doi:10.1038/nature12160

Supplementary Methods

M.1 Definition and properties of dimensionality

Consider an experiment with a discrete set of c different experimental conditions, corresponding to all distinct combinations of task-relevant variables. For example, in the experiment that we analyzed, the animal has to remember a sequence of two visual cues. The first and the second cue can be any of four visual objects, and the second cue cannot be the same as the first one, giving a total of $4 \times 3 = 12$ different types of cue combinations. Moreover, there are two possible ways of testing the memory of the animal (recognition or recall task), and hence the total c is $12 \times 2 = 24$.

We now consider the neuronal activity of N neurons in one particular time bin. Specifically, for each experimental condition we consider the vectors whose components are the mean firing rates of all neurons. These are a total of c distinct vectors, each of which is represented by a point in an N-dimensional space. We first consider the ideal noiseless case, in which the response of the neurons is exactly the same in every trial that corresponds to a particular experimental condition. In this case we define the dimensionality d of this set of points as the rank of the $N \times c$ matrix whose columns are the c vectors representing the mean firing rates in the different experimental conditions. This dimensionality is equivalent to what is technically referred to as the algebraic dimension of the vector space spanned by these points. For enough neurons ($N \geq c$), d is bounded by c, which is the maximal dimensionality. When all the N neurons encode the same task-relevant variable, the dimensionality is low because the c vectors are highly correlated. The dimensionality d can be significantly smaller than c also in the case in which each neuron encodes only one task-relevant variable or a linear combination of task-relevant variables (i.e. when there are no non-linear mixed selectivity neurons).

The dimensionality d is related to the capacity of a linear classifier that reads out the c vectors representing the mean firing rates of the N neurons. In particular, the number of possible implementable classifications grows exponentially with d (see Supplementary Section S.7). For example, when the dimensionality is maximal ($d = c$), the c vectors containing the mean firing rates are linearly independent, and this allows a linear readout to classify the inputs in all possible ways. For a binary readout that can generate only two possible outputs depending on whether the weighted sum of the inputs is above or below an activation threshold, all 2^c possible binary classifications can be implemented (i.e. there is a set of weights for all possible sets of outputs).

So far, we have considered a fictitious noiseless case. Let us now consider the situation in which the firing rates are noisy. This is clearly the situation of the experimental data, as the response of each neuron varies from trial to trial. Let us consider the case in which the neuronal response in each trial can be generated by adding independent noise to a given firing rate that depends only on the experimental condition. This firing rate is unknown and it can be determined exactly only in the ideal situation in which the experiment is repeated under the same conditions an infinite number of times. In realistic situations it is possible to estimate the firing rate by taking the average over a limited number of trials. This estimate

of course is subject to noise, and it varies when the experiment is repeated and a different set of trials is used for computing the average.

In the presence of this type of noise, the dimensionality, defined now on the estimated firing rates, is almost always maximal. As an illustrative example, consider an experiment with c different experimental conditions and in which N neurons are recorded. Let us assume now that in the noiseless case, these points lie on a line in the N-dimensional space, so that the dimensionality of the set of points is 1. When noise is added and we replace these points with those representing the noisy estimated firing rates, the points move away from the line in all directions. Even if the noise is tiny, they are now likely to span all the c dimensions (i.e. the matrix containing the c vectors representing the estimated firing rates is full rank). Unfortunately, this high-dimensionality does not correspond to an increased performance of a linear classifier. Indeed, in the noisy case one has to consider the cross-validated performance of a readout that classifies individual trials. In other words, one has to train the classifier on a set of trials, and then test the classification performance on a different set of trials, in which the realization of the noise is different. This is the typical problem that any readout in the brain has to solve. The displacements from the line due to noise are not consistent across repetitions of the same trial, and hence the geometry of the set of c points is not inherently different from the original line of the noiseless case. For this reason we devised an analysis that is described in detail below (see Supplementary Methods M.7) and it is based on the cross validated performance of a linear classifier. Our estimate of the dimensionality in the noisy case would match the dimensionality d defined above in the limit of zero noise.

The main limitation of the neural representations of pure selectivity neurons derives from their low-dimensionality. Pure selectivity cells that exclusively encode task type for instance span a one-dimensional space given by the line going through the two N-dimensional points that correspond to the average activity patterns representing the recall and the recognition task. Analogously, neurons that are purely selective to cue 1 or cue 2 span a maximal dimensionality of $4 - 1 = 3$ (i.e. the number of visual objects, which is 4, minus one). The total dimensionality that can be obtained when all these pure selectivity neurons are combined into a single population is then $3 + 3 + 1 = 7$, to which we have to add one in order to take into account the additional degree of freedom given by the displacement of the patterns of activity from the origin. This limits the number of binary classifications implementable by a linear classifier to $2^8 = 256$. Notice that the dimensionality would not change if the pure selectivity neurons are replaced with linear mixed selectivity neurons (as for Neuron 3 in Fig. 1).

This situation has to be contrasted with the maximal dimensionality afforded by the task structure, which is equal to the total number of experimental conditions (all combinations of cue 1, cue 2 and task type), i.e. $4 \times 3 \times 2 = 24$. This dimensionality can be achieved only when neurons with non-linear mixed selectivity neurons are included in the representation, and accommodates the implementation of $2^{24} \sim 16 \times 10^6$ binary linear classifications.

We restricted the analysis of the error trials to one task type, the recall task. In that case the maximal dimensionality for pure selectivity neurons that encode cue 1 and cue 2 is $3 + 3 = 6$ (to which we have to add one, again to take into account the displacement from

the origin). The maximal dimensionality is instead $3 \times 4 = 12$.

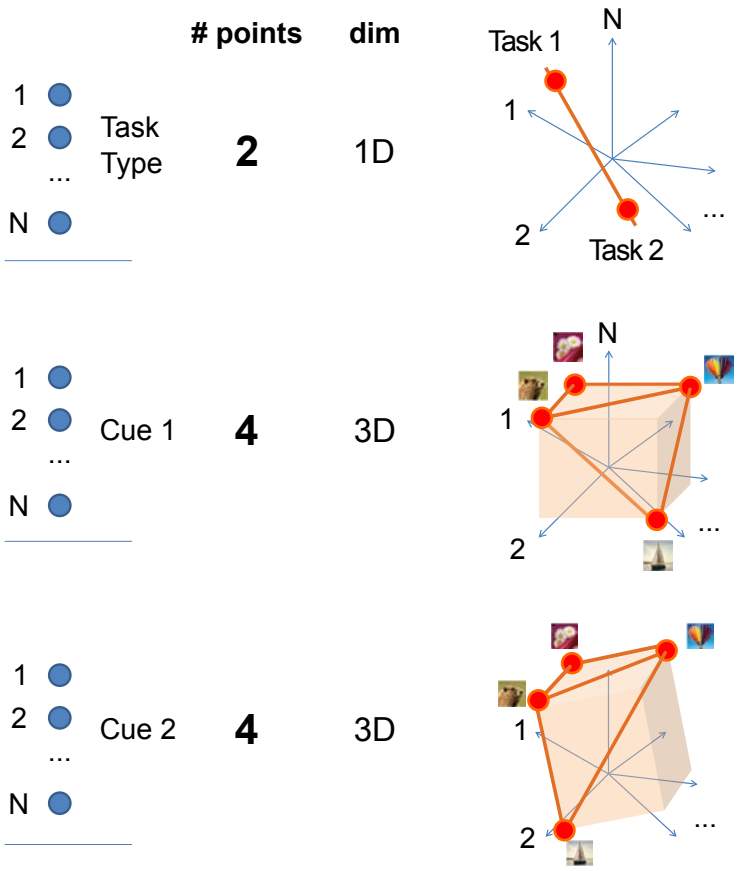

Figure M1: Expected dimensionality for neural representations with pure selectivity neurons in the memory sequence task. Filled blue circles represent neurons. Each population of neurons encode only one task-relevant aspect (pure selectivity). From the top, there are first the neurons encoding task type, then cue 1 and cue 2. Each population has size N and patterns of activity can be represented as a points in an N-dimensional space (red circles). However, as there are only two possible task types, the patterns of activity generated by recall (task 1) and recognition task (task 2) representations lie on a line (a vector space of dimensionality 1). Indeed, there is always a line going through the two points (red), regardless the specific representation of the two tasks. Analogously, the patterns of activity generated by cue 1 and cue 2 pure selectivity neurons lie in a space of dimensionality 3, as there are only 4 possible objects). The dimensionality of the neural representations then equals the dimensionality of the vectors spaces defined by the patterns of activity, to which we have to add one in order to take into account the additional degree of freedom given by the displacement of the patterns of activity from the origin.

M.2 Definition of pure, linear and non-linear mixed selectivity

The response of a neuron in a specific time interval is usually thought of as a function of the task-relevant aspects that are under experimental control (in our case, the task type and cue identity variables). We define the linear component of the firing rate response of a neuron as the part of the response that can be explained by a regression model in which the firing rate is regressed against one factor per task-relevant aspect (in addition to a bias term). According to this model, the effect on the firing rate due to a change of multiple task-relevant aspects, is described as the linear sum of the effects due to changes in each individual aspect. We then define the non-linear component as the part of the average firing rate that remains after subtracting the linear component from the neural response. Here we formally develop these definitions.

The linear model of the neuronal response

We quantify the selectivity of a neuron to the various aspects of the task in a specific time bin by fitting its firing rate FR with the following linear regression model:

$$\mathrm{FR} = \beta_0 + \sum_{i=1,2} \beta_{T,i}[\mathrm{Task} = i] + \sum_{j=\mathrm{A,B,C,D}} \beta_{C1,j}[\mathrm{Cue1} = j] + \sum_{k=\mathrm{A,B,C,D}} \beta_{C2,k}[\mathrm{Cue2} = k] + \eta, \quad \text{(M1)}$$

where the terms in square brackets represent a design matrix whose elements indicize the type of trial according to task type, cue 1 and cue 2 identity. These terms are either zero or one, depending on the trial under consideration. So for instance, [Task = 1] is equal to one when the model is fitting the firing rate in task type 1 trials. Accordingly, the sums in the equation run over the values that the aspects can acquire (task type is either 1 or 2, the identity of the cues are one of A, B, C or D). The coefficient β_0 is a bias and η represents a Gaussian noise term. Fitting the model described by equation (M1) to the firing rate is equivalent to performing a 3-way ANOVA, where the three factors correspond to task type, cue 1 identity and cue 2 identity. The βcoefficients that are determined by the fit quantify the selectivity to their corresponding aspects. Because equation (M1) does not include any interaction factors among the aspects, the model is linear.

The non-linear model of the neuronal response

A neuron's activity might not be satisfactorily fitted by the linear model of equation (M1), as the model may explain only a small fraction of the activity's variance. In particular, the linear model might not be a complete description of the detailed structure of the PCH histograms describing the average firing rate within conditions.

In order to capture the deviations of a neuron activity from the linear model, we used a different model to describe the remaining residual variance of the neurons' firing rate. As it captures the deviation of the average firing rate from linearity, this model will be used to define the non-linear mixed selectivity component of the neural responses. Specifically, after

fitting a linear model, i.e. equation (M1), to a neuron activity, we subsequently fit a 1-factor model with as many levels as the number of trial types, i.e. conditions. This will give us one coefficient per condition, which can be used to capture the deviation of the average firing rate in every condition from the linear fit. Equivalently, these coefficients are the within condition averages of the residual responses after subtracting the linear model.

In the case of the task that we are considering we first need to fit equation (M1) to obtain the βcoefficients. We then compute the residuals defined as:

$$\widehat{\mathrm{FR}} = \mathrm{FR} - \beta_0 - \sum_{i=1,2} \beta_{T,i}[\mathrm{Task} = i] - \sum_{j=\mathrm{A,B,C,D}} \beta_{C1,j}[\mathrm{Cue1} = j] - \sum_{k=\mathrm{A,B,C,D}} \beta_{C2,k}[\mathrm{Cue2} = k]. \tag{M2}$$

We then fit $\widehat{\mathrm{FR}}$ with the 1-factor model:

$$\widehat{\mathrm{FR}} = \sum_{(i,j,k)=(1,\mathrm{A,B})\ldots,(2,\mathrm{C,D})} \beta^{nlin}_{ijk}\ [\mathrm{Task} = i] \cdot [\mathrm{Cue1} = j] \cdot [\mathrm{Cue2} = k] + \eta, \tag{M3}$$

where η represents a Gaussian noise term, and the sum is over all task type, cue 1 and cue 2 identity combinations (which are 24 in our case). In this way we obtain the coefficients β^{nlin}_{ijk} that quantify the non-linear mixed selectivity component of the fitted neuron.

Definition of pure, linear and non-linear mixed selectivity components

Fitting the models that were just described to a neuron activity allows us to determine the average firing rate $\langle \mathrm{FR} \rangle_c$ in one particular time bin for every condition c of the task (c is an index that identifies a particular combination of cue 1,2 identity and task type). This procedure moreover allows us to decompose the average firing rate as the sum of two terms, a linear and a non-linear mixed selectivity component:

$$\langle \mathrm{FR} \rangle_c = \langle \mathrm{FR} \rangle^{lin}_c + \langle \mathrm{FR} \rangle^{nlin}_c.$$

The linear component is captured by the coefficients of the linear model defined above:

$$\langle \mathrm{FR} \rangle^{lin}_c = \beta_0 + \sum_{i=1,2} \beta_{T,i}[\mathrm{Task}_c = i] + \sum_{j=\mathrm{A,B,C,D}} \beta_{C1,j}[\mathrm{Cue1}_c = j] + \sum_{k=\mathrm{A,B,C,D}} \beta_{C2,k}[\mathrm{Cue2}_c = k].$$

where Task_c, $\mathrm{Cue1}_c$, $\mathrm{Cue2}_c$ indicate the particular combination of task type, and cue 1, cue 2 identity that corresponds to condition c. When the βcoefficients are significantly different from zero only for the factors of one specific aspect (cue 1, cue 2 or task type), then the neuron is said to have *pure selectivity*, in the sense that the task-relevant variables are not mixed.

The non-linear component is the average residual firing rate $\langle \mathrm{FR} \rangle^{nlin}_c = \langle \mathrm{FR} \rangle_c - \langle \mathrm{FR} \rangle^{lin}_c$, or equivalently, the firing rate obtained from the non-linear model described in the previous section:

$$\langle \mathrm{FR} \rangle_c^{nlin} = \sum_{(i,j,k)=(1,\mathrm{A},\mathrm{B})\ldots,(2,\mathrm{C},\mathrm{D})} \beta_{ijk}^{nlin} \, [\mathrm{Task}_c = i] \cdot [\mathrm{Cue1}_c = j] \cdot [\mathrm{Cue2}_c = k].$$

where again Task_c, $\mathrm{Cue1}_c$, $\mathrm{Cue2}_c$ indicate the particular combination of task type and cue 1, cue 2 identity that corresponds to condition c. Notice, that the sum will single out exactly one coefficient per condition c.

Sparse coding neurons: 'grandmother cell'-type activity

Our definition of non-linear mixed selectivity includes the particular type of neurons denoted as 'grandmother cells', neurons whose response is exclusively triggered by highly specific stimulus compounds. In the context of the experiment we are examining, a 'grandmother cell' would correspond to a neuron that is selective to a unique condition, specified as a single task type, cue 1 identity and cue 2 identity. Such a 'grandmother cell' would in other words be active only for one of the 24 possible trial conditions, corresponding to a coding level of $1/24$.

The activity of a 'grandmother cell' would not be effectively described by the pure or linear mixed selectivity model in equation (M1), since this model describes the effect of the individual presented cues or the task type variable. It would however be modeled in a straightforward way as a non-linear mixed selectivity cell as in equation (M3), simply by setting to zero all but one specific β_{ijk}^{nlin} coefficient corresponding to the condition the cell is selective to.

M.3 The procedure for removing classical selectivity

We first describe how to remove the classical selectivity to task type from neuron i. Neuron i is said to have classical selectivity to task type (for instance) at time t whenever the average activity measured at time t during recognition task (task type 1) is significantly different from the average activity measured at time t during the recall task (task type 2). More formally:

$$\langle \nu_i^{c'}(t) \rangle_{c' \in T_1} \neq \langle \nu_i^{c''}(t) \rangle_{c'' \in T_2},$$

where the averages $\langle \cdot \rangle_{c \in T_{1,2}}$ indicate computing the mean activity over conditions in which task type is Task 1 and Task 2, respectively, and $\neq$ means statistically significantly different. A task type selective neuron is shown in Fig. M2a. The activity of the neuron is shown as a function of time and different colors correspond to different combinations of task type and cue 1 identity (the identity of cue 2 is ignored). The right panel of Fig. M2a shows PCHs that illustrate this activity specifically during the 100 ms indicated by the yellow arrow in the left panel. The activity averaged over Task 1 trials is significantly different from the activity averaged over Task 2 trials ($p < 10^{-10}$, two-sample t-test) as indicated by the dashed lines indicating the average activity levels. Notice that, apart from this difference, individual bars of the PCH can also be different within the conditions of the same Task type (mixed

selectivity). The idea behind the procedure to remove classical selectivity is motivated by the question whether these individual differences in activity during same task trials are important for the neural code, and in particular whether they can encode information in addition to what is encoded by neural selectivity.

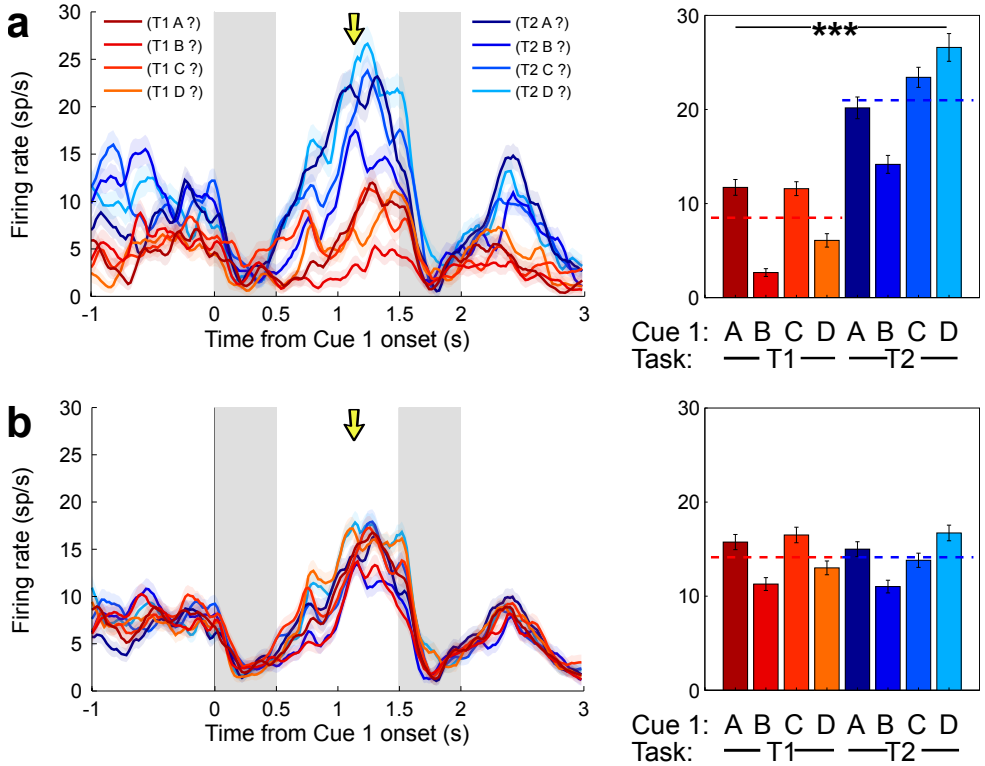

Figure M2: Procedure for removing classical selectivity illustrated on the task type factor. **a**, The PSTH and the PCH of a selective neuron ($p < 0.001$, two-sample t-test) are shown (the PCH is computed at the time indicated by the yellow arrow, i.e. the time bin starting one second after the onset of cue 1). The PCH conditions are sorted so that all task type 1 conditions are on the left (reds), whereas task type 2 conditions are on the right (blues). For each task type 1 trial, we add the firing rate of a randomly chosen task type 2 trial, and then we divide by 2, computing the average between the two firing rates. The trial can belong to any condition of task type 2 and basically acts as noise that has the same statistics as the task type 2 trials. The procedure is repeated for task type 2 trials by adding randomly chosen task type 1 trials. **b**, Result of the equalization procedure to eliminate selectivity with respect to task type. The left panel shows that the average activity after the equalization procedure no longer displays a preference for task type 2 trials. The right panel shows that the PCH still maintains some information about the specific task type-cue 1 combination, despite the average selectivity to task-type being erased.

We removed the classical selectivity of neuron i in Fig. M2a by contaminating the spikes recorded during one task type with spikes recorded during the other task. To be more specific, every time we sample a spike count from the trials in condition $c' \in T_1$ we superimpose an additional noise source by adding a spike count sampled from a randomly selected task type

2 trial recorded in the same time bin but in a random condition $c'' \in T_2$. The spike counts that we obtain as a result of this generation process are realizations of a random variable that, for a specific condition c' in Task 1, is formally denoted as:

$$\widetilde{\nu}_i^{c'}(t) + \widetilde{\Delta}_i^{c'}(t) = \nu_i^{c'}(t) + \Delta_i^{c'}(t) + \sum_c \gamma_{T_2}(c)\left(\nu_i^c(t) + \Delta_i^c(t)\right),$$

where $\gamma_{T_2}(c)$ indicates a random variable that selects at random a condition c'' for which the task type is Task 2, i.e. $\gamma_{T_2}(c'') = 1$ for only one $c'' \in T_2$ and $\gamma_{T_2}(c) = 0$ for $c \neq c''$. Analogously, we add a random spike count from a random trial recorded in Task 1 blocks every time we sample from a condition c'' in which the Task is of type 2:

$$\widetilde{\nu}_i^{c''}(t) + \widetilde{\Delta}_i^{c''}(t) = \nu_i^{c''}(t) + \Delta_i^{c''}(t) + \sum_c \gamma_{T_1}(c)\left(\nu_i^c(t) + \Delta_i^c(t)\right).$$

After this procedure, the average firing rates during the two tasks is exactly equalized:

$$\langle \widetilde{\nu}_i^{c'}(t)\rangle_{c' \in T_1} = \langle \widetilde{\nu}_i^{c''}(t)\rangle_{c'' \in T_2}.$$

We illustrate this equalization procedure on the 100 ms time bin around 1.2 s after Cue 1 onset indicated by the yellow line in the left panel of Fig. M2a whose corresponding PCH is illustrated in the right panel. In this time bin the neuron is strongly selective to task 2 ($p < 10^{-10}$, two-sample t-test).

Fig. M2b shows the corresponding PSTHs and PCHs after applying the equalization procedure. The equalization got rid of the task selectivity and the only information that the neuron is codifying is now in the pattern of mixed selectivity present in the differences between PCHs for different conditions.

To remove the classical selectivity to cues from neuron i we proceed in an analogous way, with the only difference that the end result is that the neuron firing rate does not exhibit any preference for any of the 4 cues, while for removing the classical selectivity to task we only needed to consider 2 task types. Because of this difference, as in the previous equalization equation every time we sample a spike from a condition c' corresponding to cue C_1 we simply add a noise term $\sum_c \gamma_{C_i}(c)\left(\nu_i^c(t) + \Delta_i^c(t)\right)$ for every one of the other cues C_i, $i = 2, 3, 4$. This procedure has as a result that

$$\langle \widetilde{\nu}_i^c(t)\rangle_{c \in C_1} = \langle \widetilde{\nu}_i^{c'}(t)\rangle_{c' \in C_2} = \langle \widetilde{\nu}_i^{c''}(t)\rangle_{c'' \in C_3} = \langle \widetilde{\nu}_i^{c'''}(t)\rangle_{c''' \in C_4},$$

i.e. the average activity is the same for all of the 4 cues, as desired. Notice that in this case the noise introduced by the described procedure is even higher than in the case of task type selectivity.

In Supplementary Section S.5 we show that once classical selectivity to a specific aspect is removed, that individual aspect can only be read out with a non-linear decoder (provided that, as we verified, non-linear mixed selectivity encodes it). In a sense this is the converse result of one of our main conclusions: in the same way non-linear mixed selectivity is essential to linearly decode mixtures of task-relevant aspects, pure selectivity is essential to linearly

decode individual aspects.

M.4 Generation of pure selectivity neurons from data

Fig. 4 compares the number of binary classifications that can be implemented by the neurons recorded in PFC against the number of binary classifications of ideal selective neurons. Ideal neurons that are selective to a particular task aspect were generated from the recorded data in order to preserve the noise statistics of the original recorded neurons. Specifically, we computed the spike counts in 100 ms time bins for all conditions. We then sorted them according to the spike count, and assigned the highest spike counts to the "preferred" conditions of the aspect and the lowest spike counts to the unpreferred ones (e.g. in the case of task type selectivity the 12 conditions with the highest spike count were assigned to task type 1 and the others to task type 2). Fig. M3 illustrates this operations on a recorded neuron which is non-selective to task, and it is transformed into a neuron which displays a strong preference to Task 1. Fig. M3a shows the PCHs of the considered neuron in a 100 ms around 2.1 s after Cue 1 presentation. The neuron does not display any statistically significant selectivity to task type ($p > 0.4$, two-sample t-test). Fig. M3b shows the reordered histograms. The spike counts belonging to the highest histograms are then assigned to the preferred task, and the others are assigned to the unpreferred task (Fig. M3c). The resulting PCH correspond to the activity of a neuron which is very strongly selective to Task 1 in the considered time bin ($p < 0.001$, two-sample t-test) and, by construction has the same noise statistics as the original recorded neuron.

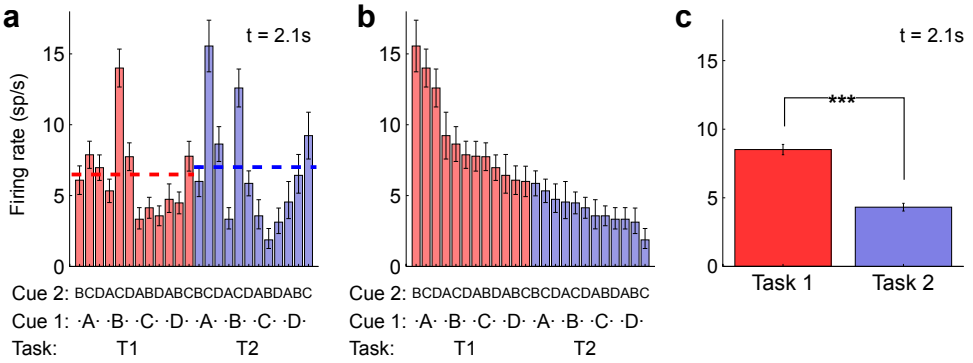

Figure M3: Creating pure selectivity neuron from a non-selective neuron. **a**, PCHs of the non-selective neuron during the 100 ms time bin at 2.1 s after the presentation of Cue 1. Averaging across conditions within the same task type block does not reveal any difference across blocks. Notice however that this neuron displays some mixed selectivity to Cue 1, Cue 2 and Task. **b**, The spike count histograms are sorted in decreasing order and the highest half is assigned to the preferred Task, while the lowest half is assigned to the unpreferred task. **c**, As a result of this way of resorting the spike counts we get a neuron which is strongly selective to Task 1 ($p < 0.001$, two-sample t-test).

Fig. M4 shows the result of applying this procedure on all time bins for the neuron

considered in Fig. M3. The left panel of Fig. M4a is the PSTH resulting from sorting the trials according to Task Type. The right panel shows the PCHs specifically during the time bin pinpointed by the yellow arrow in the left panel (the same as Fig. M3a). The left panel of Fig. M4b shows the PSTH of the resulting task selective neuron, and the right panel focuses on the time bin indicated by the yellow arrow (the same as Fig. M3c).

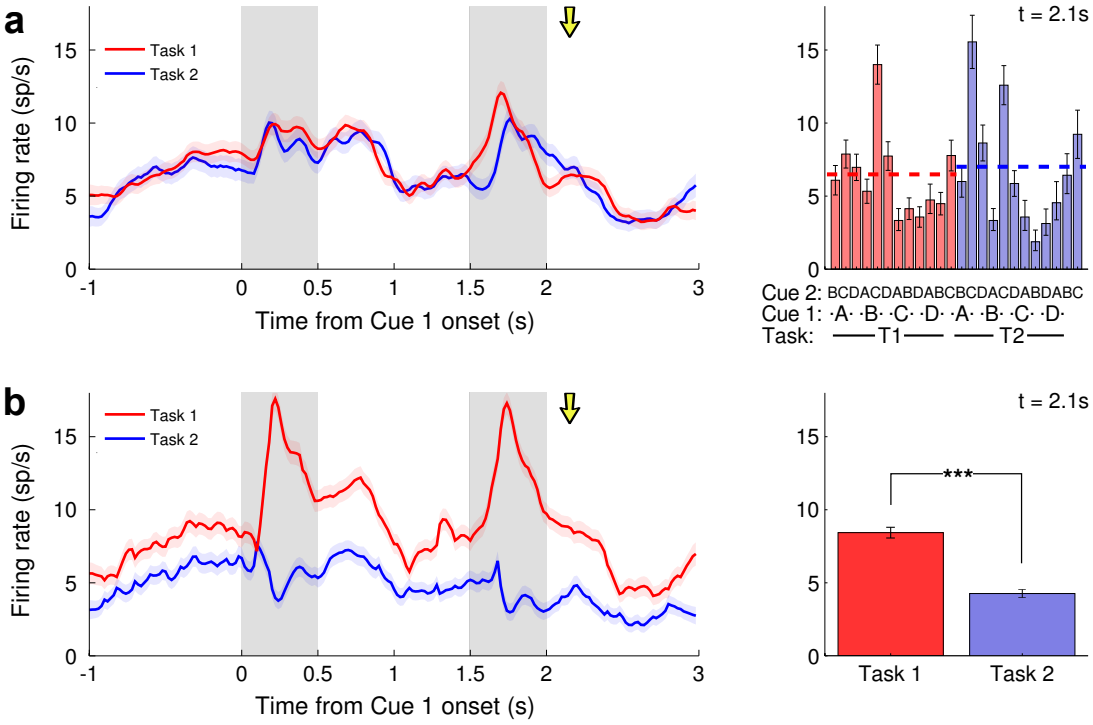

Figure M4: Creating pure selectivity neuron from a non-selective neuron. **a**, Activity of a recorded neuron obtained by sorting the trials according to task type (left panel). This neuron does not exhibit any statistically significant selectivity to task type. In an example 100 ms time bin at time 2.1 s after Cue 1 presentation (yellow arrow in left panel) the PCH (right panel) shows mixed selectivity to combination of Cue 1, Cue 2 and Task type, despite the fact that the overall averages across task type do not denote any selectivity to task. **b**, The activity of an artificial neuron selective to task 1 obtained from the original recorded neuron in **a**. The right panel shows the PCHs in a 100 ms time bin around 2.1 s after Cue 1 presentation. In that time bin the artificial neuron is strongly selective to Task type 1 ($p < 0.001$, two-sample t-test) and the distribution of spike counts across trials is by construction identical to the one of the original recorded non-selective neuron.

Fig. M5 shows the result of applying the decoding multi-layer decoding algorithm on a population of artificial pure selective neurons. The population is composed of three sub-populations of neurons, each encoding a different task aspect. The first sub-population encodes task type (25 neurons prefer task type 1 and 25 prefer task type 2), the second one, cue 1 (25 neurons per visual object, for a total of 100 neurons) and the third one encodes cue 2 (again, 25 neurons per visual object). For this relatively small population we can decode

all task-relevant aspects with very high accuracy, indicating the the neurons are strongly selective to task types.

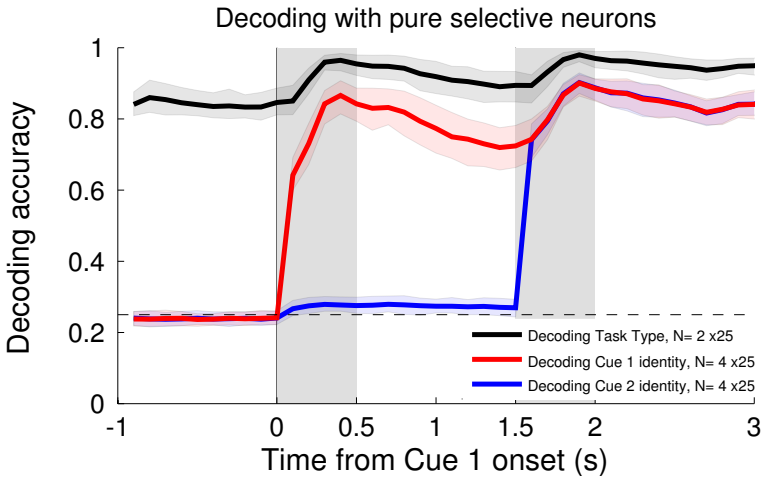

Figure M5: Reading out from pure selectivity neurons. The graph shows the performance of the multi-layer decoding algorithm applied on a population of pure selectivity neurons. The population consists of 25 neurons per value of all aspects of the task.

M.5 The non-linear population decoder

Architecture and training of the multi-layer population decoder. The decoding algorithm used for Figs 3e,f is based on a multi-layer network architecture. For every 200 ms time bin in the task we train a multilayer-network to solve the multiclass learning problem of associating a pattern of activity recorded during any one of the 24 task conditions to the corresponding condition. The network consists of an ensemble of weak classifiers that are combined in a error-correcting output code scheme to decode the condition as illustrated in Fig. M6 (see also [24]).

The training procedure is done by using the trials in the training set to estimate the average activity patterns $\underline{\nu}^c(t)$ for every condition c. Every weak classifier in the ensemble is assigned a pair of conditions c' and c'' and is trained to maximally discriminate the corresponding average population patterns $\underline{\nu}^{c'}(t)$ and $\underline{\nu}^{c''}(t)$. For all other conditions the weak classifier is not trained. In other words, every readout specializes on a particular pair of conditions, and error-correcting output code scheme takes care of combining these specialized competences in a global classifier [24].

Training is formulated as a constrained quadratic programming problem (see [25, 26]), an optimization method that is used to improve generalization performance in the Structural Risk Minimization and Support Vector Machines literature [11] and is equivalent to a maximal margin perceptron algorithm [23, 27, 26].

The resulting 24×23 weak classifiers (as said, one for every pair of conditions) are combined with an appropriate code matrix to output the condition in the output layer [24].

To improve the temporal stability and generalization performance of the final decoder, we bootstrap aggregate it with 2 additional decoders [28] obtained by performing the training procedure of data 100 ms earlier and later in the trial with respect to the center of the original 200 ms interval.

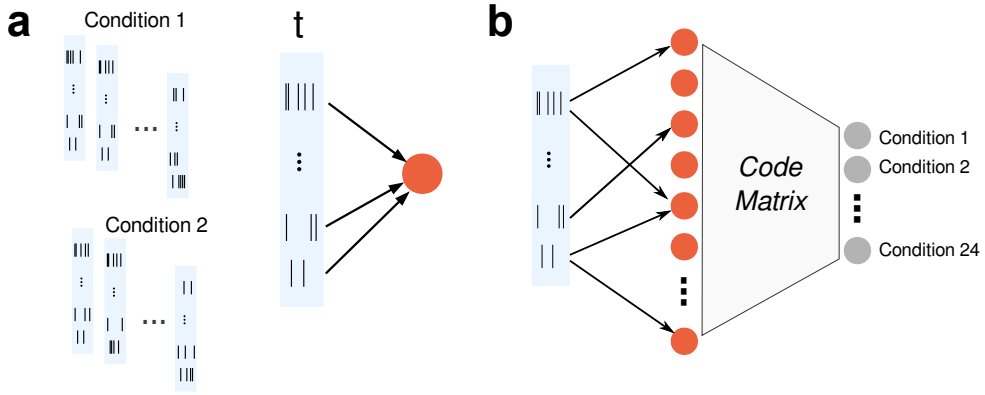

Figure M6: Training of the multi-layer population decoder. **a**, For every time bin t and every pair of conditions a weak classifier is trained to discriminate between the two classes with a maximal margin algorithm applied on estimated average population activity patterns $\underline{\nu}^c(t)$ corresponding to the two conditions. **b**, The weak classifiers that discriminate between all of the condition pairs are combined with an appropriate error-correcting output code matrix to output the most likely condition.

Population activity patterns. As explained in the previous paragraph, our decoding algorithm works by an error-correcting output code scheme [24] that pools together a set of weak-classifiers trained on pairs of task conditions. The activity patterns were obtained as pseudo-simultaneous population response vectors [15]. In particular:

- Given a condition c ($c = 1, ..., 24$) at time t, the activity of a neuron i is assumed to be a stochastic variable $\nu_i^c(t) + \Delta_i^c(t)$, where $\nu_i^c(t)$ indicates the instantaneous average firing rate that is supposed to be consistent across trials, while $\Delta_i^c(t)$ is a noise term that is randomly sampled from a noise distribution at every trial. Such quantities are estimated from the data by calculating the spike counts in 200 ms windows around time t. Spike counts are whitened before being used to train the decoder. We however verified that this does not change the decoder's performance.

- For every neuron we divide trials into a training and a test set. The test set is used to cross-validate the performance of the decoder. The test set contains a number of trials that is the same for all conditions, and is determined as one quarter of the number of trials of the condition with the smallest number of trials. Training trials and testing trials are whitened separately.

- The patterns for training and testing are generated as follows (see Fig. M7). Given condition c, for every neuron i we pick a trial at random n_i^c either in the training or the test set (depending on whether we want to train or test the population decoder), and we use these trials to generate a pattern of population activity $\underline{\nu}^{\underline{n}^c}(t)$, where $\underline{n}^c = (n_1^c\ n_2^c \ldots n_i^c \ldots n_N^c)$ indicates a trial per neuron. This procedure is repeated for each condition.

- As detailed in the next section, training is performed using an estimate of the average activity patterns $\underline{\nu}^c(t)$ for every given condition computed on the trials belonging to the training set.

- The performance of the population decoder is assessed by cross-validation on activity patterns obtained by repeatedly resampling trials from the test set to generate activity patterns corresponding to all the conditions that have to be discriminated. We think of this as a way to probe the generalization performance of the population decoder in the face of the the trial-to-trial noise $\Delta_i^c(t)$.

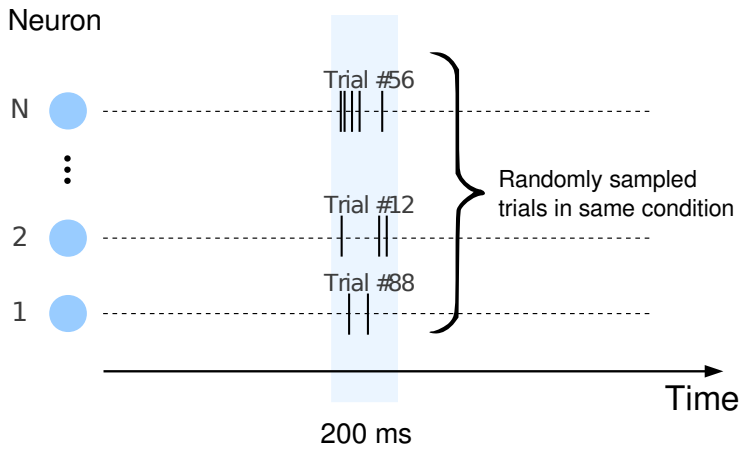

Figure M7: Generation of pseudo-simultaneous population activity patterns used to train and cross validate the population decoder. We generate a pattern of activity corresponding to a given condition c by randomly sampling for every neuron i a trial n_i^c recorded in condition c. We compute the spike count in such trial in 200 ms bins.

M.6 Extrapolating the decoding accuracy and dimensionality for a large number of input neurons.

In order to extrapolate the decoding accuracy curves to more neurons than the 237 that were actually recorded, we generated a set of new neurons obtained by resampling the recorded neurons with a different assignment of the cue identity labels. For example, the labels of the four stimuli A,B,C,D can be permuted into B,D,C,A. This means that the recorded activity

in condition $(A, B, 1)$ (first cue, second cue, task type), will be assigned to condition $(B, D, 1)$ for the response of the new neuron. this procedure can be repeated for every one of the way of permuting the four-stimulus sequence A,B,C,D to generate up to 24 new neurons per recorded neuron. We applied this procedure to the 185 neurons for which we had recorded at least 8 trials per condition, generating in this way a maximum of $185 \cdot 24 = 4440$ neurons.

Notice that the assignment of the cue labels is anyway arbitrary as in different experimental sessions different stimuli were used. Because we assume that the visual objects used as cues in the experiment are statistically equivalent in the neuronal response they evoke, we do not expect that the statistical properties of the neural representations (including dimensionality) are modified by the operation of relabeling. We however verify that this is indeed the case by estimating the number N_c of classifications implementable by a linear classifier reading out the 185 recorded neurons, and comparing this with the result obtained from the extrapolation procedure applied to a random subset of 50 neurons extrapolated to 185 neurons (see Supplementary Section S.8).

M.7 Computing the number of implementable binary classifications, N_c, and its relation to dimensionality

Here we describe the procedure for estimating N_c, the number of implementable binary classifications in a specific time bin. This quantity depends on the number N of neurons the firing rate vectors are generated from. However, as we will discuss and show with simulations in Supplementary Section S.7, under mild conditions on the covariance of the neural activity, the ratio between N_c and the total number of possible binary classifications T_c approaches an asymptotic value as N increases. In the same Section we also show that this asymptotic value is robustly related to the dimensionality of the neural representations through a logarithmic operation.

Given c different task conditions (c ranges from 1 to 24 in the sequence memory task), we can in principle generate c classes of activity patterns. Intuitively, the highest dimensionality corresponds to the situation where any of the c classes corresponds to a distinct average activity pattern, and the noise of the neural discharge is low enough to still allow for discrimination. A way to assess whether the patterns are discriminable is to arbitrarily assign them to one of two classes and check whether there exists a linear classifier that can separate the two classes. Every assignment of the patterns to the two classes corresponds to a so-called "coloration", since every pattern is fictitiously associated to one of two possible colors [29]. Accordingly, the linear classifier is supposed to assign the positive output $+1$ to the patterns in the first class, and the negative output -1 to the patterns in the second class.

It is well-known from Cover's theorem [29, 26] (see also Supplementary Section S.7) that the existence of such a classifier depends on how many activity patterns c we try to classify. For c below a critical value c^* the classifier exists with high probability, irrespective of the coloration it has to implement, while for a c above the critical value c^* it is very unlikely that the corresponding coloration can be implemented by a linear classifier. This phase transition

becomes increasingly sharp as the critical value c^* increases.

For a given pair of c and N, we followed this procedure to estimate the number of implementable classifications:

- We generate an N-dimensional average activity vector for each of c conditions from the training set of trials. Each component of the activity vector corresponds to the firing rate of a neuron averaged over a specific time bin and across trials within the same condition. For the main analysis in Fig. 4 we considered for instance the 800 ms time bin in the middle of the first and the second-object delay period.

- Each generated vector can be associated to either ± 1 (the desired response of the linear classifier), for a total of $T_c = 2^c$ possible binary classifications. We use quadratic programming to learn all these associations (or randomly sample 100000, if $2^c > 100000$) maximizing the learning margin (see [25, 26]).

- We then quantify the number of binary classifications that can be implemented given a value of c. We call this number n_c (in lowercase letters). Notice that n_c is a function of c, and its sum over c equals the total number of implementable classifications N_c, the quantity we are after. The number n_c is computed by cross-validation. This is done by estimating the cross-validation error of the trained classifications on a test set of patterns generated with a hold-out set of trials. We choose a threshold $\theta_{error} = 0.2 - 0.25$ on the allowed maximal cross-validation error, meaning that binary classifications with a generalization performance of at least 75-80% are counted as implementable, while those with a lower performance are counted as non-implementable. The analysis is robust with respect to the choice of θ_{error}, as its precise value only rescales the noise, meaning that it does not influence the asymptotic value that we will obtain for N_c as N increases (see Supplementary Section S.7)

Fig. M8 shows the result of this process for different neural populations. Fig. M8a considers a population of pure selectivity neurons generated as in Supplementary Methods M.4. Every curve is the fraction of implementable binary classifications plotted against the number of considered conditions c. Different curves are generated with a different number N of neurons. Darker lines correspond to larger N. The continuous lines are obtained by fitting the data points with the function $y(c) = exp(-c^{\alpha}/\beta)$. The curves were obtained by fitting $log(y(c))$ in log-log space, where it appears as a linear curve. The dashed lines around the fits are the curves generated from the 95% confidence bounds on the estimated coefficients assuming Gaussian error around the linear fit.

Fig. M8b shows analogous plots obtained on the neural population of recorded PFC neurons extrapolated by reshuffling the cue identity labels as explained in Supplementary Methods M.6.

A general trend that we note from Fig. M8 is that as the number N of neurons in the population increases (i.e. for darker curves), the number of binary classifications that can be implemented also increases.

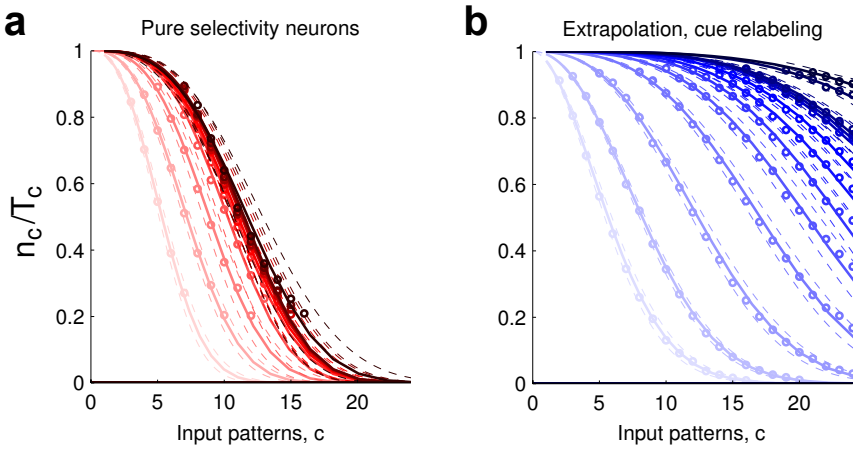

Figure M8: Fraction of implementable binary classifications n_c/T_c as a function of the number of input patterns c among the possible 24 conditions that are classified. Different curves correspond to different number of neurons representing the inputs (darker curves correspond to increasing number of neurons). The firing rate activity of the neurons is computed over the 800 ms in the middle of the second-object delay period. **a**, Fraction of implementable binary classifications n_c/T_c with a test error of at most $\theta_{error} = 0.2$. The neural population over which the inputs are sampled are pure selectivity neurons generated as explained in Supplementary Methods M.4. **b**, Same as in **a** for recorded PFC data after extrapolation to larger number of neurons by relabeling object identities as explained in Supplementary Methods M.6. Data points are fitted with the Weibull curves $y(c) = exp(-c^{\alpha}/\beta)$.

As hinted at before, the fraction of implementable classifications n_c/T_c undergoes a phase transition as c approaches a critical value c^*: for values of c below c^* an arbitrary binary classification can be implemented as a linear separator with probability close to 1, while above c^* this probability sharply drops close to 0. Because of the robust character of this transition, it is convenient to quantify the total number of implementable classifications by first determining c^*, and declaring that the implementable classifications are those with a number of inputs c that is smaller than c^*. Once we know c^* we can then compute the total number of implementable binary classifications simply by counting the number of binary classification with up to c^* inputs, which gives $N_c = 2^{c^*}$. In Supplementary Section S.7 we moreover show that c^* corresponds to the dimensionality d.

We now determine c^* through the plots exemplified by Fig. M8. The critical value c^* is defined as the maximal value of c for which for which almost all of the binary classifications out of the total T_c can be implemented. We therefore determine it by choosing an arbitrary threshold sufficiently above chance level (we chose 95%) and stipulating that c^* is the first value of c for which the fraction n_c/T_c falls below it. The advantage of this approach is that, because of the universal character of the phase transition, our analysis is robust with respect the choice of the threshold. In fact, for c^* sufficiently large, all thresholds become equivalent, as the curves n_c/T_c against c approach a step function. In addition we verified that in situations where we can theoretically compute the dimensionality in an independent

manner, i.e. in simulations (see Supplementary Section S.7) and for the pure selectivity neurons, the theoretical values and the analysis results are in good agreement. This is in general a good criterion for choosing the threshold.

The confidence intervals in our estimate of c^* are obtained through the confidence intervals of the fit of the n_c/T_c as a function of c. The relation $N_c = 2^{c^*}$ then allows us to compute N_c and relative confidence intervals.

So far we have considered the scenario where the output of the binary classifications is unconstrained, a situation that exponentially favors "dense" classifications with roughly as many positive as negative outputs. In Supplementary Section S.12 we show that as the output coding level of the implemented binary classifications decreases, so does the advantage of dense high-dimensional representations over low-dimensional pure-selectivity representations. However, the coding level for which the two kinds of representations achieve comparable performance decreases exponentially as the number of task conditions increases. Consequently, already at modest task complexity involving tens of conditions, the output coding level for which low-dimensional representations reach a capacity that is comparable to high-dimensional ones is too low to be physiologically or computationally relevant.

M.8 The contribution of linear and non-linear mixed selectivity components to the collapse in dimensionality observed in error trials

We estimated the number of implementable classifications during the recall task, separately for correct and error trials. Fig. 5a shows the results of this analysis and reveals that the dimensionality collapses in the error trials.

In Figs 5c,d we tease apart the contributions of linear and non-linear mixed selectivity components of the recorded neurons. First, in Fig. 5c we plotted the number of implementable classifications as a function the input neurons when the linear mixed selectivity component is removed, leaving only the non-linear mixed selectivity component (see Supplementary Methods M.2). Specifically, for each neuron, we subtracted from the firing rate FR of each trial the linear mixed selectivity component:

$$\mathrm{FR}^{nonlin} = \mathrm{FR} - \langle \mathrm{FR} \rangle_c^{lin},$$

where $\langle \mathrm{FR} \rangle_c^{lin}$ is the linear component of the mixed selectivity and it is defined in Supplementary Methods M.2. We then performed the analysis described in Supplementary Methods M.7 to determine the number of implementable classifications.

Analogously, in Fig. 5d, we computed the number of implementable classifications when the non-linear mixed selectivity component is removed:

$$\mathrm{FR}^{lin} = \mathrm{FR} - \langle \mathrm{FR} \rangle_c^{nonlin},$$

where $\langle FR \rangle_c^{lin}$ was defined in Supplementary Methods M.2.

Supplementary Sections

S.1 The limitations of neural representations based on pure selectivity

QUESTION: The functional relevance of high-dimensional representations generated by non-linear mixed selectivity stems from the large number of diverse response functions that a downstream linear readout can implement (what in the manuscript is called the number of implementable binary classifications). Why is it important to be able to implement a large number of response functions? How is it possible that a neural representation contains all the information about the task-relevant variables and still there are response functions that cannot be implemented?
ANSWER: It is important to distinguish between the information contained by population activity patterns, and the format in which such information is represented. In order to implement a particular behavior it is typically not sufficient that the neural representations encode the relevant variables. These variables might additionally need to be encoded in a way that allows downstream neural circuits to implement specific responses. The type of responses that are implementable depends on the specific readout mechanisms. In our manuscript we assume as customary that readouts are linear classifiers. In this section we summarize previous results that derive from this assumption and show why it is important to implement a large number of response functions. We describe a simple example that illustrates that the neural implementation of a task typically requires more response functions than those that represent the individual task-relevant variables.

The execution of a task cannot be reduced to a set of simple "static" input-output relations. The neural circuits have to be designed so that the recurrent dynamics generates the correct output at the right time. In sequential tasks this requires to preserve the memory of events which are separated in time, over timescales that can be significantly longer than the inherent time constants of individual neurons. This is one simple argument as to why individual neurons have sometimes to implement input-output functions that may not be directly related to the structure of the behavioral task. For instance, in [9], the work that inspired this analysis, we built a general class of models of working memory, in which the neural circuit is designed to have a set of stable states, each encoding the relevant pieces of information for reacting properly to future events. Every sensory stimulus is thought of as an event that induces transitions between states. These states are represented by stable patterns of reverberating activity. In this model, not only is it important to tune the synaptic weights so that the sensory stimuli steer the neural activity from one state to another, but it is also crucial to stabilize the patterns of reverberating activity. This necessity introduces a series of additional mathematical constraints that depend on the architecture and the dynamics of the neural circuits, and that are equivalent to imposing that each neuron implements a particular set of input-output functions. For example, in order to have a stationary pattern of activity, each neuron has to persistently generate a specific output for the input representing the stable pattern of activity (see also the Discussion).

We now consider a simple example in which it is not sufficient to encode all the task-relevant variables separately, but it is necessary to mix them non-linearly. This limitation is a consequence of the simplicity of the specific readout mechanism that we are considering. Indeed, for simple linear readout neurons there are input-output relations that cannot be implemented even in the case in which all the task-relevant variables are correctly encoded in the input. These variables have to be represented in a format that is suitable for being processed by the specific readout that we are considering. Here we describe an illustrative example related to our task in which it is important to have non-linear mixed selectivity neurons in the input.

Let us take a constructionist view and consider how to build a neural circuit that executes the recognition sequence memory task. In particular, let us think of the conditions that such a circuit needs to satisfy in order to generate a differential activity for match and non-match sequences. Let us for the moment restrict our considerations to two particular cue sequences. In the first, the sample sequence A→C is followed by A as first test cue. This sets the neural circuit in a state that we arbitrarily call S_{ACA}. In the second, the sample sequence A→D is followed by A as first test cue, setting the circuit in state S_{ADA}. In these two states the identity of the second test cue will determine whether the test sequence is a match or a non-match. A neuron signalling the eventual result should respond differently to match and non-match trials. For example, let us assume that the neuron is active to signal match. When the circuit is in state S_{ACA}, such a neuron should be activated by the presentation of C as second test cue, while it should be inactive for D. Conversely, when the circuit is in state S_{ADA}, it would be active at the presentation of D as second test cue, and it would be inactive for C.

We consider the case where the neural population only consists of pure selectivity neurons that encode either the state or the sensory input, and show that, under this constraint, it is not possible to find synaptic weights such that the readout neuron implements the required match/non-match selectivity. Notice that this will be true despite the fact that all task-relevant variables (the states and the sensory input) are represented across the population of pure selectivity neurons.

The activity of the four possible kinds of pure selectivity neurons (two to encode the state and two for the sensory input) is illustrated in the following table, where every row corresponds to a neuron, every column corresponds to a trial condition, and the entries indicate whether the neuron responds (1) or not (0) in a specific condition. The last row of the table indicates the desired output of the readout neuron as a function of condition (+ indicating that the total synaptic input to the neuron exceeds the activation threshold, - indicating that it does not). The input I_{out} to the readout neuron is now a sum of the activities of the pure selectivity neurons, weighted by a corresponding synaptic weight:

$$I_{out} = w_{S_{\mathrm{ACA}}}\ \mathrm{FR}_{S_{\mathrm{ACA}}} + w_{S_{\mathrm{ADA}}}\ \mathrm{FR}_{S_{\mathrm{ADA}}} + w_{\mathrm{C}}\ \mathrm{FR}_{\mathrm{C}} + w_{\mathrm{D}}\ \mathrm{FR}_{\mathrm{D}}.$$

We now have to find synaptic weights w_i such that I_{out} exceeds the activation threshold θ in conditions $S_{\mathrm{ACA}} + \mathrm{C}$, and $S_{\mathrm{ADA}} + \mathrm{D}$ (matching), and remains below threshold for conditions $S_{\mathrm{ACA}} + \mathrm{D}$ and $S_{\mathrm{ADA}} + \mathrm{C}$. These requirements result in the four inequalities (S1)-(S4).

neuron	$S_{\mathrm{ACA}}+\mathrm{C}$	$S_{\mathrm{ACA}}+\mathrm{D}$	$S_{\mathrm{ADA}}+\mathrm{C}$	$S_{\mathrm{ADA}}+\mathrm{D}$
$\mathrm{FR}_{S_{\mathrm{ACA}}}$	1	1	0	0
$\mathrm{FR}_{S_{\mathrm{ADA}}}$	0	0	1	1
FR_{C}	1	0	1	0
FR_{D}	0	1	0	1
output	+	-	-	+

$$I_{out} = w_{S_{\mathrm{ACA}}} + w_{\mathrm{C}} > \theta \quad \text{(S1)}$$
$$I_{out} = w_{S_{\mathrm{ACA}}} + w_{\mathrm{D}} \leq \theta \quad \text{(S2)}$$
$$I_{out} = w_{S_{\mathrm{ADA}}} + w_{\mathrm{C}} \leq \theta \quad \text{(S3)}$$
$$I_{out} = w_{S_{\mathrm{ADA}}} + w_{\mathrm{D}} > \theta \quad \text{(S4)}$$

Summing inequalities (S1) and (S4) gives $w_{S_{\mathrm{ACA}}} + w_{S_{\mathrm{ADA}}} + w_{\mathrm{C}} + w_{\mathrm{C}} > 2\theta$, while summing inequalities (S2) and (S3) gives $w_{S_{\mathrm{ACA}}} + w_{S_{\mathrm{ADA}}} + w_{\mathrm{C}} + w_{\mathrm{C}} \leq 2\theta$. These two conditions are obviously contradictory, which means that no set of weights w_i and threshold θ can implement the desired output. This example is actually equivalent to the "exclusive OR problem" [30], which is well known and it has been widely studied. Notice that a readout neuron cannot discriminate between match or non-match, but it can certainly be trained to respond differently to states or to sensory inputs.

While simple and illustrative, the previous example is actually representative of the non-linear separability problems that arise with low-dimensional representations due to pure selectivity (see e.g. [9]). As we described in the paper, non-linear mixed selectivity generates high-dimensionality at the level of population activity patterns. High-dimensional activity patterns can be flexibly read out, meaning that they can generate a wide repertoire of readout responses, that can in turn be responsible for the initiation of relevant behavioral output. What we hypothesize is that this capability is what occasionally goes awry, giving rise to error trials.

S.2 Definition of mixed selectivity index and distribution of mixed selectivity across the recorded cells

QUESTION: The linear and non-linear mixed selectivity components are defined in Supplementary Methods M.2. How are these components of the single cell activity distributed across the recorded neurons?
ANSWER: We can define an index that quantifies the relative contribution of these linear and non-linear components in explaining the variance of every neuron's activity. Here we report the distribution of this mixed selectivity index and quantify the importance of the non-linear mixed selectivity component across the population of cells.

We apply the procedures explained in Supplementary Methods M.2 to all neurons in our dataset to quantify the linear and the non-linear selectivity components. We find that 47 cells, corresponding to 19.8% of the 237 recorded cells, show statistically a significant non-linear mixed selectivity component during the two-object delay epoch (ANOVA, $p<0.05$). This is a remarkable fraction of the population, considering for instance that during the two-object delay epoch 41% of the recorded cells show a statistically significant selectivity to the task type factor and 59% of the cells show object-selectivity [3].

To further quantify non-linear mixed selectivity at the population level we devised a **non-linear mixed selectivity index**, I_{nlin} defined as follows:

$$I_{nlin} = \frac{Var_{nlin} - Var_{lin}}{Var_{nlin} + Var_{lin}}, \tag{S5}$$

where Var_{nlin} is the total variance of the activity explained by the non-linear mixed selectivity component, while Var_{lin} is the total variance explained by the linear mixed selectivity component (M1). This index is normalized between -1 and 1. A positive index denotes a neuron that is better characterized as a non-linear mixed selectivity neuron, while a negative index corresponds to a mostly linear mixed selectivity neuron. Fig. S1 plots the distribution of non-linear mixed selectivity indices across the population of neurons. The median of the distribution is negative (-0.15), indicating that overall the population has a stronger linear mixed selectivity component. However, a considerable part of the population (80 neurons, corresponding to 34% of the recorded cells) shows a positive index, indicating that a considerable fraction of neurons are more accurately described as non-linear mixed selectivity neurons.

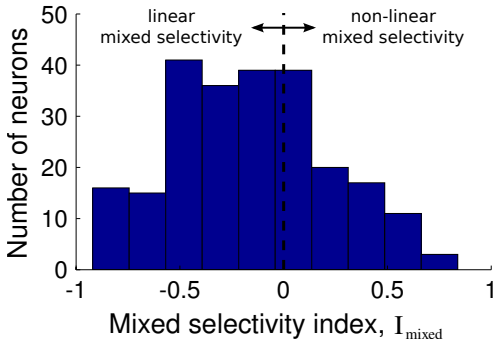

Figure S1: Distribution of non-linear mixed selectivity index (equation (S5)) across the population of recorded neurons during the two-object delay epoch. The dotted line indicates an index equals to zero and demarcates the distinction between mostly linear mixed selectivity (negative index) and mostly non-linear mixed selectivity neurons (positive index). A fraction corresponding to 34% percent of the neurons shows positive non-linear mixed selectivity index.

S.3 Contribution of linear and non-linear mixed selectivity to population decoding

QUESTION: The methods developed in Supplementary Methods M.2 allow us to tease apart the linear and non-linear mixed selectivity components of a cell's activity. How do these components individually contribute to the decoding performance of task-relevant information at the population level?
ANSWER: For every neuron we can extract the linear and non-linear components of the activity as explained in Supplementary Methods M.2. We can then in turn remove one of

these components from every neuron and apply our population decoding method to read out the task-relevant variables. As expected, removing the non-linear mixed selectivity component does not strongly impact the decoding performance, indicating that the linear mixed selectivity component robustly encodes the task-relevant aspects. The remarkable result is that removing the linear mixed selectivity component encoding a specific task-relevant aspect affects the readout accuracy of the removed aspect, but does not completely ablate it. This confirms that, at the population level, non-linear mixed selectivity can encode information that is absent at the single neuron level.

Fig. S2 shows the result of applying the decoder to the population of 237 recorded cells, after removing from each cell the contributions to the firing rate that was fitted by the non-linear mixed selectivity model. The decoding accuracy is essentially unaffected by the operation, consistent with the idea that linear mixed selectivity is sufficient if we only want to separately decode individual task aspects.

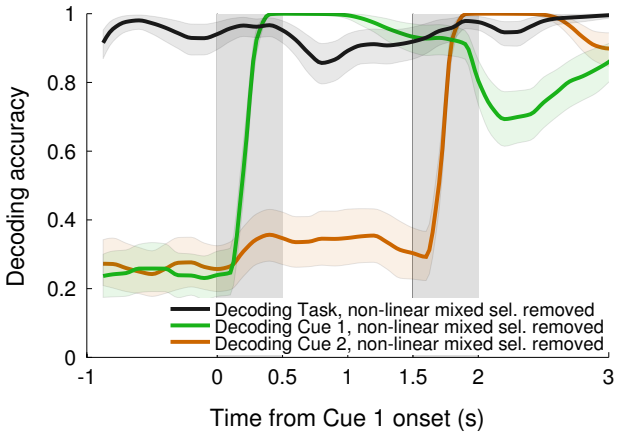

Figure S2: Decoding from the recorded 237 neurons after removing the non-linear mixed selectivity components. Decoding performance is unaltered in accordance with the notion that pure selectivity is sufficient to separately encode individual task-relevant aspects (see also Supplementary Methods M.4, where we generate pure selectivity neurons and decode from them).

Fig. S3 shows the result of instead removing from every cell the individual linear mixed selectivity components. We decode a task-relevant aspect after the coefficients that correspond to that specific aspect have been subtracted from the activity. So for instance we fit the model described by equation (M1) to every neuron, remove from the activity the coefficients that quantify the selectivity to task-type, and then train our population decoder to read out task type. We then apply the same procedure to all other task-relevant aspects. Decoding performance is strongly impacted, but nonetheless remains above chance level. Remarkably, mixed selectivity can be exploited to read out individual task-relevant aspects from the neural population, despite none of the neurons being individually selective to them. Moreover, as the trial progresses and mixed selectivity becomes increasingly important, the decoding performance progressively increases.

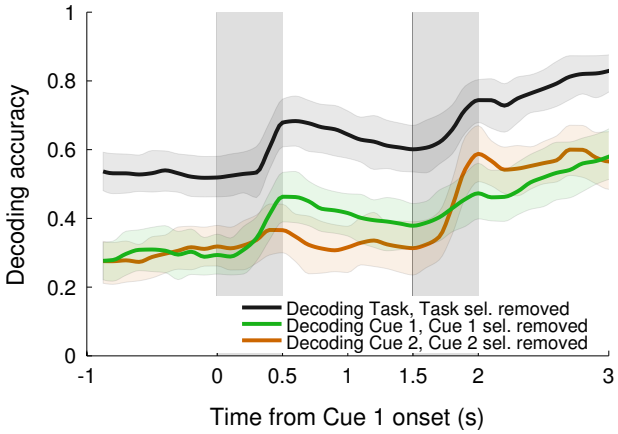

Figure S3: Decoding from the recorded 237 neurons after removing the individual linear mixed selectivity factors. Decoding performance decreases considerably, but still remains above chance level. Remarkably, mixed selectivity can be exploited to read out individual task-relevant aspects from the collective neural population, despite none of the neurons being individually selective to them in the conventional sense. Moreover, as the trial progresses and mixed selectivity becomes increasingly important, the decoding performance progressively increases. This is in accordance with Fig. 3f that shows that this results still hold if selectivity is removed by equalizing all moments of the firing rate distribution, and not only the average firing rate. See also Supplementary Methods M.3.

The analysis that is illustrated in Fig. S3 shows that information about the individual task-relevant aspects can still be collectively decoded from the neural population, even if the activity of individual cells is not on average selective to them. In Supplementary Methods M.3 we illustrate a more radical method for removing selectivity that not only equalizes the average firing rate across trials, but actually equalizes all moments of the firing rate distribution. This method also has the additional advantage that it does not need to rely on the sequence of linear regressions illustrated in Methods M.2. Fig. 3f shows that even equalizing all moments of the distributions of firing rates, mixed selectivity can still be exploited to readout individual task-relevant aspects.

S.4 Population decoding on simultaneously recorded spiking activity

QUESTION: Are temporal correlations between the activity of recorded neurons important for the decoding performance?
ANSWER: The temporal correlations from the simultaneously recorded neurons (on average around 5-6 neurons were simultaneously recorded in a session) do not affect the performance of the decoders.

The dataset that we analyzed comprised multiple simultaneously recorded neurons. On average more than 5 neurons were recorded simultaneously in a single session. This in

principle allows us to address the question whether temporal correlations between the firing rates of different neurons can affect the decoding accuracy. These correlations could have either a positive or a detrimental effect on the accuracy [31, 32]. Here we show that taking these correlations into account in the measure that the data allow us to, does not have any measurable impact on the decoding accuracy.

Fig. S4 shows the decoding performance traces of the multi-layer decoder used in Fig. 3e (continuous lines) overlaid with the performance traces obtained by taking into account the correlations of simultaneously recorded firing activity (Fig. S4, dashed lines). We took into account the correlations in the variability of the neuronal firing rates by generating activity vectors that contain the firing rates of the same trials for neurons that were recorded simultaneously. This modification of both the training and testing procedure of the decoder does not affect the final performance.

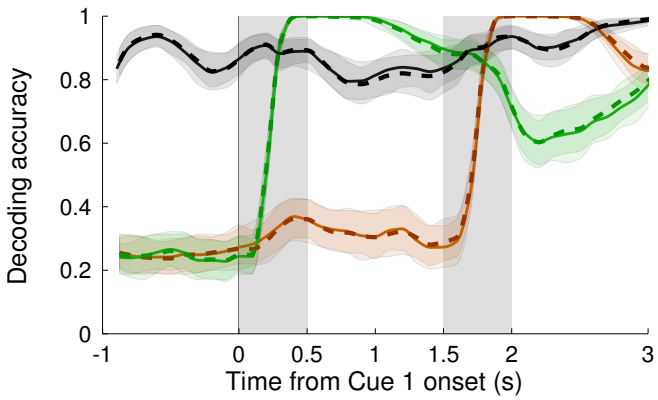

Figure S4: Temporal correlation of simultaneously recorded neurons does not have any detectable effect on the decoding performance of the multi-layer decoder. We reproduce Fig. 3e showing the performance traces of the multi-layer decoder as a function of time (continuous lines), overlaid with the performance of an analogous decoder applied on spike-count histograms obtained by taking into account the simultaneity of the activity of neurons recorded in the same session (dashed lines). On average more than 5 neurons were simultaneously recorded in every session. The two decoding accuracy curves coincide, meaning that the performance of the readout is unaffected by the correlations in the variabilities of neuronal firing rates of the simultaneously recorded neurons.

S.5 Non-linear separability of equalized neural populations

QUESTION: Why did we have to use a non-linear multi-layer decoder when selectivity was eliminated?
ANSWER: Our manipulation for removing selectivity makes the neural representations non-linearly separable. A non-linear decoder is then required.

Here we show that the vectors composed of the firing rates of neurons whose average activity is equalized across conditions are always non-linearly separable. In other words, our

procedure for removing selectivity makes the neural representations artificially non-linearly separable and hence requires more complex non-linear readouts.

We assume that p conditions belong to context C_1 and p different conditions belong to context C_2. The activity of a given neuron i in a trial in condition c is denoted by ν_i^c. Its average over trials is denoted by $\langle \nu_i^c \rangle$.

The assumption that none of the neurons shows a differential activity for context C_1 or C_2 is formulated as follows:

$$\sum_{c_1 \in C_1} \langle \nu_i^{c_1} \rangle = \sum_{c_2 \in C_2} \langle \nu_i^{c_2} \rangle. \tag{S6}$$

A linear readout discriminating between context C_1 and context C_2 consists in a set of weights w_i and a threshold θ such that:

$$\langle \sum_i w_i \nu_i^{c_1} \rangle > \theta, \text{ for all } c_1 \in C_1, \tag{S7}$$

$$\langle \sum_i w_i \nu_i^{c_2} \rangle < \theta, \text{ for all } c_2 \in C_2. \tag{S8}$$

It is now straightforward to show that assumption (S6) is in contradiction with the previous two equations, demonstrating the main claim. Summing equations (S7,S8) over conditions and using (S6) we get

$$p\ \theta \overset{(S7)}{<} \sum_{c_1 \in C_1} \langle \sum_i w_i \nu_i^{c_1} \rangle = \sum_i w_i \sum_{c_1 \in C_1} \langle \nu_i^{c_1} \rangle \overset{(S6)}{=} \sum_i w_i \sum_{c_2 \in C_2} \langle \nu_i^{c_2} \rangle \overset{(S8)}{<} p\ \theta,$$

which cannot be satisfied for any threshold θ.

S.6 Decoding the ordered sequence of the presented objects

QUESTION: The identity of the first and second sample cues are encoded in the recorded activity. Is the information about their order of presentation also present in the neural activity?
ANSWER: Yes, the sequence of visual objects and their presentation order can be accurately decoded.

The multi-layer population decoder can be used to simultaneously decode both the identity and the order of the two presented sample objects (object sequence). Fig. S5 shows the decoding accuracy for the object sequence as a function of time for 237 recorded neurons (lighter trace) and the extrapolated performance for 1000 neurons (darker trace). The 237 recorded neurons are enough to get a decoding performance that peaks above 80% during the presentation of the second cue. With 1000 neurons the decoding performance gets very close to 100% at the end of the second delay period.

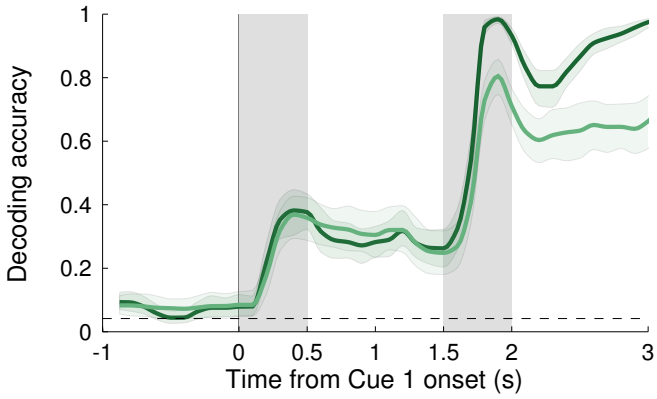

Figure S5: Performance of the multi-layer population decoder in decoding the sequence of presented objects (same format as Fig. 3e). The lighter line shows the decoding accuracy reading out from the 237 recorded neurons. The darker line shows the extrapolation for 1000 neurons. Shaded areas indicate 95% confidence intervals.

S.7 Relation between input vectors dimensionality, noise and number of implementable classifications

QUESTION 1: How does the number N_c of implementable classifications depend on the dimensionality of the neural representations?
ANSWER: N_c grows exponentially with the dimensionality of the neural representations.

QUESTION 2: How is our estimate of dimensionality based on the number of implementable classifications affected by noise?
ANSWER: Our estimate is robust to noise. The number of implementable classifications grows with the number of neurons in the neural representations that are classified, reaching an asymptote that only depends on the dimensionality of the neural representations. The speed of convergence to the asymptote depends on noise, but the value of the asymptote, and hence our estimate of dimensionality, does not.

In linear classification problems the dimensionality of the input space determines the number of implementable classifications N_c in the noiseless case (see e.g. [26]). For this reason it is possible to determine the dimensionality of the input by estimating N_c. However, in the presence of noise the relation between dimensionality and N_c is not obvious. We will show in this section that when the number of noisy neurons representing the input increases, N_c keeps increasing and eventually reaches an asymptote. This is true as long as the principal components of the noise covariance matrix that are aligned along the weights of the decoder are negligible (see e.g. [32]), which is the case that we consider here. This asymptote is directly related to the dimensionality of the input.

Consistently to what we expect from theoretical considerations [26], we show through simulations that:

- The number N_c of binary classifications (i.e. the number of "colorations" of the inputs, where colors refer to the desired output for each input) that can be implemented is robustly related to the dimensionality of the vector space spanned by the input patterns. In particular, in the noiseless case, this quantity is independent from the number of neurons representing the inputs.

- In the noisy case, increasing the number of input neurons decreases the noise, and allows for a larger number of implementable binary classifications N_c. This number eventually saturates to the noiseless limit independently of the noise amplitude.

We illustrate these points in a simulation of a classification task that has a similar structure as the original sequence learning task executed by the monkeys. We consider an hypothetical task which comprises 8 different binary relevant aspects. The task is therefore defined over $2^8 = 256$ possible conditions. We create neural populations that represent these conditions with activity patterns of increasingly higher dimensionality d, where the dimensionality of a set of patterns is defined as the rank of the vector space generated by the linear span of the patterns. What we do to increase the dimensionality of the patterns is to progressively add neurons whose activity is determined by random combinations of the value of an increasing number of aspects. The more aspects jointly determine the activity of a neuron, the higher will be the dimensionality of the ensuing patterns (see e.g. [9, 20]). We then compute the number of binary classifications that can be implemented on these representations, as we did for Figs M8 and 4a,b.

Fig. S6a shows the analogous graph as Fig. M8a plotting the fraction of correctly implementable binary classifications as a function of the number of inputs on which they are computed. This is done for representations with dimensionality of $d = 13, 20, 35$, respectively, a fixed level of noise such that the signal-to-noise ratio (both in training and testing) for any individual neuron is $SNR \approx 10$, and an increasing number of neurons (darker curves correspond to more neurons). Fig. S6b is analogous to Figs 4a,b for the three populations of Fig. S6a and two levels of noise during testing ($SNR \approx 100$ and $SNR \approx 10$). The figure illustrates that the noise level only determines how many neurons are required to reach saturation, while the saturation level only depends on the dimensionality d of the patterns. Fig. S6c shows the fraction of implementable binary classifications, N_c, for different inputs and different dimensionalities at saturation, that is for noise approaching zero or, equivalently, number of neurons approaching infinity. These curves allow us to plot the number of implementable binary classifications N_c as a function of input dimensionality d in Fig. S6d. From the figure we can see that N_c grows exponentially with the dimensionality of the input representations.

S.8 The procedure for extrapolating the performance of the linear classifier does not change the number of implementable classifications

QUESTION: The main result of the article is that neural representations in PFC are high-

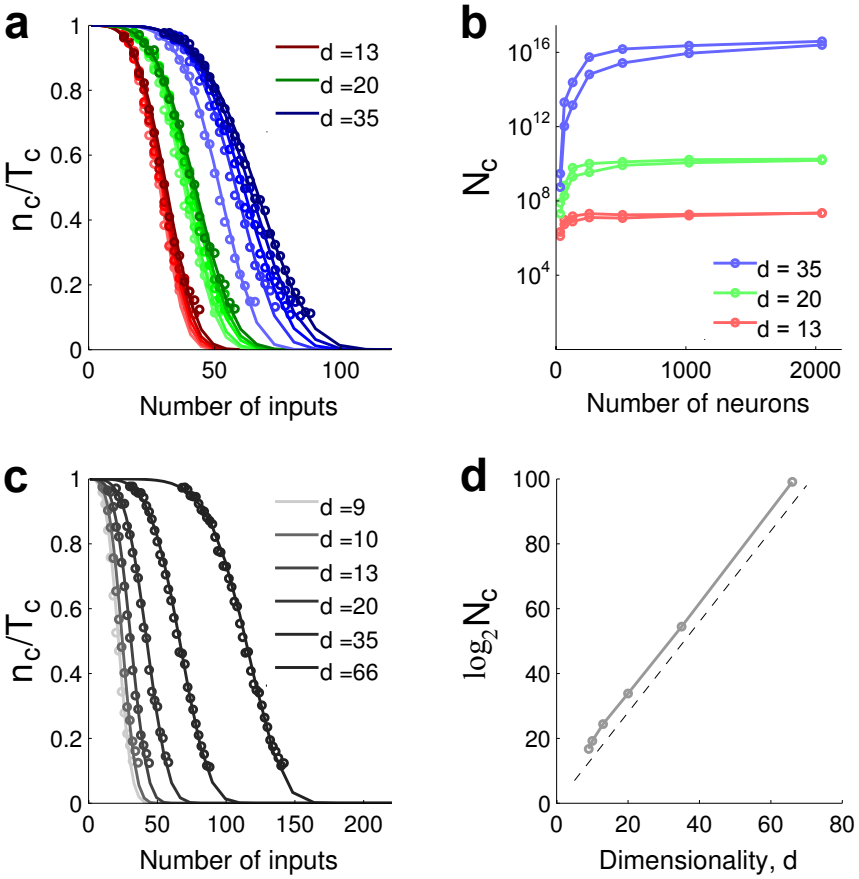

Figure S6: Relation between input vectors dimensionality and number of implementable classifications. **a**, Fraction of correctly implementable binary classifications as a function of the number of input patterns for different dimensionalities ($d = 13, 20, 35$) and different number of neurons (darker curves correspond to more neurons). **b**, The advantage of increasing the number of neurons saturates very quickly and at a level that does not depend on the noise intensity. The saturation level only depends on the dimensionality of the inputs d. **c**, Same as **a**, but at saturation (i.e. with enough neurons to reach saturation), for different input dimensionalities. **d**, Total number of implementable binary classifications N_c as a function of input dimensionality. N_c grows exponentially with d. The dashed line correspond to an exponential growth with exponent of 1.4, slightly lower than the theoretical one of 2 obtained in the noiseless uncorrelated case [33].

dimensional. As the representations are noisy, we needed to extrapolate the number of implementable classifications for a number of neurons that is larger than the number of recorded cells. Are these additional neurons changing artificially the number of implementable classifications?
ANSWER: The additional neurons are generated from the recorded cells by relabeling the visual objects (see Supplementary Methods M.6). We verified that this procedure does not change the number of implementable classifications.

Fig. 4 shows the curves representing the number N_c of implementable classifications as a function of the number N of neurons in the input. The number of recorded neurons for which we had enough trials to carry out the analysis is 185, but it is clear that for this value of N the curve representing N_c has still not reached an asymptote. As the dimensionality of the neural representations depends on the asymptote, we extrapolated N_c for a larger N by introducing additional neurons whose activity was generated from the recorded neuronal activity as described in Supplementary Methods M.6. We basically resampled the same recorded neurons with replacement with different permutations of the labels of the visual objects.

The introduction of new neurons with shuffled labels can potentially modify the statistics of the mixed selectivity that is present in the neural representations, and hence the value of N_c. For example it may introduce mixed selectivity to combinations of task-relevant aspects that were not present in the original representations, and this would increase the rate at which N_c increases with N. Here we verify that this is not the case. We selected a random subset of 50 recorded neurons and we extrapolated the number of implementable binary classification N_c up to 185 neurons (the number of recorded neurons for which we had enough trials to carry out the analysis). The value of N_c estimated from a subset of 50 recorded neurons and extrapolated to higher number of neurons is shown in Fig. S7 for different numbers of extrapolated neurons (black points). These points are compared with the values of N_c obtained using only recorded neurons, up to the 185 recorded neurons (green points). Fig. S7 shows that the N_c extrapolated from 50 recorded neurons coincides with the N_c obtained using only effectively recorded neurons, all the way up to a value of 185 neurons. This means that the extrapolation procedure generates estimates of N_c that grow with N at the same rate as the recorded neurons.

Unfortunately we cannot make this comparison for the asymptotic values of N_c. However, it is unlikely that the extrapolated N_c and the measured N_c coincide up to $N = 185$ and then they diverge reaching different asymptotes. This would a situation that requires: 1) a fine tuning of the balance between the noise and the amount of mixed selectivity added by the neurons with shuffled labels 2) strong and systematic asymmetries between the neural representations of different visual objects.

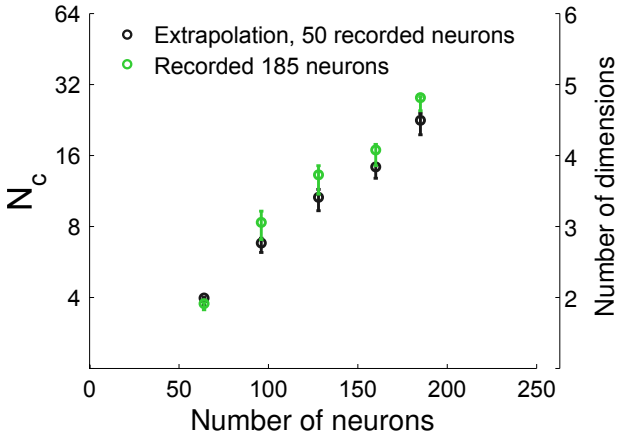

Figure S7: Verification of the extrapolation procedure. We apply the extrapolation procedure of Supplementary Methods M.6 to a random subset of 50 recorded neurons to extrapolate N_c to populations of 64, 128, 160, 185 neurons (green points). We compare the result of the extrapolation with a measure of N_c obtained with effectively recorded neurons (black points). The fact that the points overlap indicates an excellent agreement of the extrapolation procedure, with the measure obtained with actually recorded neurons.

S.9 Flexibility: execution of a new virtual task

QUESTION: Given that the neural representations are high-dimensional, is it possible to train a simulated neuron to perform a novel virtual task?
ANSWER: Yes, a simple linear classifier can be successfully trained to perform a complex novel task similar to the Wisconsin Card Sorting Task.

High-dimensional neural representations allow for a large number of possible implementable binary classifications N_c, and hence they can generate as many different responses in down-stream neural circuits. In other words, a hypothetical down-stream neuron (modeled as a linear classifier) reading out high-dimensional neural representations can be set to implement any response function corresponding to one of the N_c classifications by appropriately choosing the synaptic weights.

Here we show how an hypothetical downstream neuron receiving afferent inputs from the recorded PFC neurons can be used to implement a specific, new virtual task. The task we consider is essentially a two-dimensional Wisconsin Card Sorting Task [34, 2, 9] where the object to be considered changes depending on the task to be executed. The rule in our virtual task is essentially to either look at the first or at the second object in the sequence depending on task type, and ignore the other one. The virtual subject then has to release a bar when the relevant object is either A or B, and hold the bar when the relevant object is either C or D. The task is illustrated in detail in Fig. S8.

Fig. S9 shows the result of training an hypothetical down-stream neuron to execute the task, i.e. to respond when the agent is supposed to release the bar, and be silent when the agent has to hold the bar. The task can be directly cast as a binary classification problem and

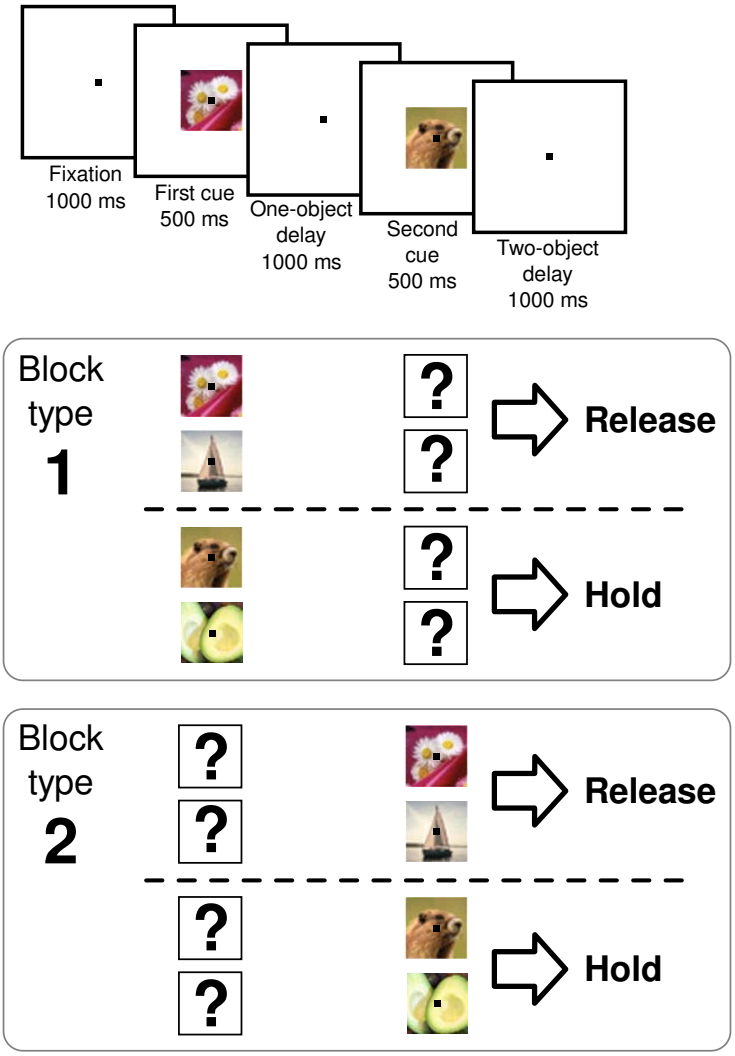

Figure S8: A hypothetical virtual task that uses the same elements of the sequence learning task. The task type variable of the original task is used to encode the currently active rule (look at cue 1 and ignore cue 2, or look at cue 2 and ignore cue 1). If the relevant cue is either object A or B the subject then has to release a bar, and is required to hold the bar if the relevant cue is either object C or D.

the down-stream neuron can be thought of as a readout neuron implementing the classification. The black curve in Fig. S9 shows that a down-stream neuron receiving input from the recorded PFC neuron can be trained to execute the task with very high accuracy. The performance exceeds 90% accuracy with around 1000 input neurons (extrapolated performance) and keeps growing as more neurons are employed. The gray curve in the figure quantifies the performance of an analogous neuron trained on the pure selectivity representations. The performance in this case saturates sightly above 75%. This curves reproduce the general trend illustrated in Figs 4a,b, where we showed that an arbitrary binary function is very

likely to be implementable when the input are generated from the recorded PFC neurons, while it is not the case when the input are generated by a population of pure selectivity neurons.

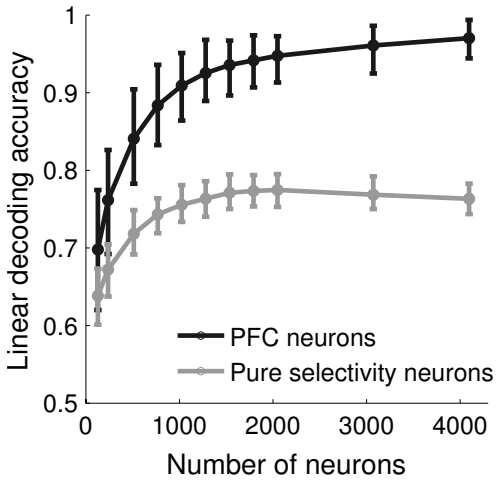

Figure S9: Cross-validation performance of a linear decoder trained on a classification task that is equivalent to the virtual task illustrated in Fig. S8. Training such a decoder can be thought of as training a neuronal response function to execute the motor output that is necessary for the task. The curves show the cross-validated performance as a function of the number of neurons for two types of representations of the input. The black curve indicate the performance when the input population is composed of the recorded PFC neurons, while the gray curve shows the case where the input population is composed of pure selectivity neurons. Consistently with what shown in Figs 4a,b, the performance of the binary function on the pure selectivity neurons is significantly worse than when the input is represented by the recorded data.

S.10 Principal Component Analysis

QUESTION: We estimated the dimensionality of the neural representation with a novel technique. Would an estimate based on Principal Component Analysis produce the same results?
ANSWER: The use of PCA to compute the dimensionality of noisy vectors requires a model of the noise distribution. This allows for an estimate of the number of principal components that are due to noise, which in turn can be used to compute a lower bound on the number of significant principal components (see [15]). This number can be used to estimate a lower bound on the number of implementable binary classifications through exponentiation. On the other hand, our analysis directly provides us with a quantification of the number of implementable binary classifications. Moreover, because our analysis is the result of a cross-validation procedure, it does not require to know and specify a model for noise variability.

An alternative approach to that of directly quantifying the number N_c of implementable

binary classifications is to look at the dimensionality of the space spanned by the inputs. This can be done by determining the number of principal components that describe the population activity vectors as was for instance done in [15] using Principal Component Analysis (PCA). PCA is a linear dimensionality reduction method that is often used as visualization tool [35]. In our case we want to determine the number of vectors that we need to linearly reconstruct all the activity patterns. A crucial step in this procedure is to determine how many of the linear components of the activity patterns are due to finite sampling of the noise. This requires some assumptions about the neural noise. We employ here the techniques developed and illustrated in [15] that allow us to estimate a lower bound on the number of significant principal components. The result is compatible with our analysis and reveals that the recorded neural activity is described by a number of principal components which is significantly larger than what would be expected from a population of pure selectivity neurons.

The first step of the technique developed by [15] is to compute the covariance matrix of the signal and the noise in the data (using a slightly different notation from [15]):

$$C_{ij} = \langle (\bar{r}_i(c) - \langle \bar{r}_i(c) \rangle)(\bar{r}_j(c) - \langle \bar{r}_j(c) \rangle) \rangle \,, \tag{S9}$$

where $\bar{r}_i(c)$ denotes the trial-averaged firing rate of neuron i in experimental condition c, and the angular brackets denote averaging over conditions.

Because the variance of the average firing rates $\bar{r}_i(c)$ can be due to both, task-relevant changes and finite sampling noise in the firing rate estimates, the authors of [15] propose a method to quantify the contribution from the noise. This gives a way to estimate which principal components of the covariance matrix C_{ij} are indicative of significant task-relevant changes in the population activity. The number of these significant PCs is in turn an estimate of the dimensionality of the activity.

Fig. S10 shows the estimated upper bound on the number of PCs needed to linearly reconstruct the activity patterns computed in a 800 ms time bin in the middle of the two-object delay epoch. Fig. S10a compares the number of significant PCs that span the activity of a population of pure selectivity neurons generated as in M.4 (gray curve) with the number of PCs that span the recorded neural activity (black curve). The gray curve saturates at 7, while the black curve shows that the recorded neural activity, for a population of 4000 neurons, is bounded from below by 17. Taken together with the considerations that we developed in Supplementary Section S.7 about the relation between the dimensionality of the input and the number of implementable binary classifications N_c, these observations are agreement with what we show in Fig. 4b.

Fig. S10b shows the analogous graphs comparing the estimated upper bound on the number of significant PCs during the recall task in correct (black curve) and error trials (gray curve). Again, consistently with our analysis on the number of implementable binary classification N_c (Fig. 5a), the Principal Component Analysis suggests that the number of PCs in the activity patterns recorded in correct trials is significantly higher than in error trials.

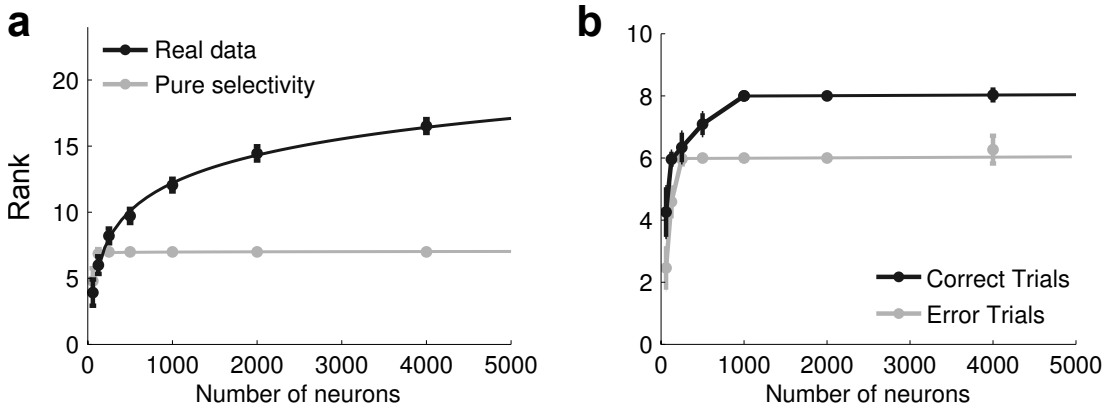

Figure S10: Number of significant principal components estimated as in [15] during a 800 ms time bin in the middle of the two-object delay epoch. **a**, Number of significant principal components (PCs) of the recorded PFC neural activity (black line), and the generated pure selectivity neurons (gray line). The recorded neural patterns display a dimensionality that is significantly larger than what would be expected under the assumption of the activity being primarily determined by pure selectivity to the aspects of the task, consistently with what we show in Fig. 4b. **b**, Number of significant PCs of the recorded PFC neural activity during the recall task for correct (black line) and error trials (gray line). Consistently with what we show in Fig. 5a the number of PCs during error trials is significantly lower than in correct trials.

S.11 Coding level of recorded PFC representations

QUESTION: The selectivity of the neurons in a neural pool is related to its coding level, a quantity that crucially determines metabolically and computationally relevant quantities. What is the coding level of the recorded representations?
ANSWER: We report a rather dense (i.e. not sparse) measured coding level.

We compute the sparseness of the recorded representations in the way that was proposed in [36] for visual images, and from that we derive the corresponding coding level f_{VG}. We will also compute the coding level in an alternative way as illustrated in [37] by counting the fraction of responses above half of the maximal mean response. This will be indicated as $f_{1/2}$. The relation between the mentioned approaches and ours is that, instead of focusing on the fraction of visual images that elicit a response and defining that as coding level, we count the number of task conditions to which every neuron is responsive.

The sparseness distribution across neurons and time bins during the recognition and recall tasks is plotted in Fig. S11. For 200 ms time bins the median Vinje-Gallant sparseness corresponds to a coding level of $f_{VG} = 0.66$. The coding level measured by counting the fraction of responses above half of the maximal mean response [37] gives an average coding level of $f_{1/2} = 0.31$. Increasing the time bin results in a larger coding level for both measures (not shown). Notice that for symmetry reasons we will be interested in the deviations of the coding level from 0.5, irrespective of whether the deviation is towards higher of lower values. In other words, a coding level of f is equivalent to a coding level of $1 - f$, because

we can reverse the role of 0's and 1's. This means that both estimates of the coding level correspond to a dense value of $f \approx 0.31 - 0.34$.

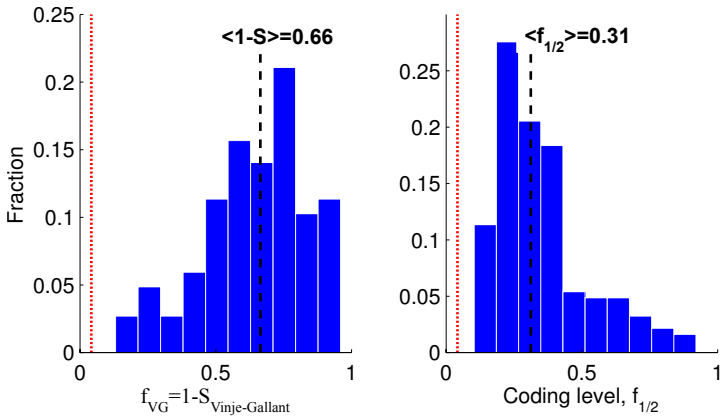

Figure S11: Coding level in PFC computed in non-overlapping 200 ms time bins during both the recognition and the recall task as in [36] (left panel) and by counting the fraction of responses above half of the maximal mean response [37]. The median Vinje-Gallant sparseness corresponds to a coding level of $f_{VG} = 0.66$ (dashed black line, left panel). The coding level measured by counting the fraction of responses above the mean response [37] gives a median coding level of $f_{1/2} = 0.31$ (dashed black line, right panel). For both measures, all of the recorded neurons have a coding level above the grandmother cell level of $1/24$ (red dotted lines).

S.12 Number of implementable classifications for sparse output coding levels

QUESTION: The recorded dense high-dimensional representations have a huge computational advantage over low-dimensional representations based on highly specialized pure-selectivity cells in terms of the number of binary classification that they allow to implement. Is this advantage preserved if we only want to implement sparse binary classifications corresponding to output neural representations with low coding level?
ANSWER: As the output coding level of the implemented binary classifications decreases, so does the advantage of dense high-dimensional representations over low-dimensional pure-selectivity representations. However, the coding level for which the two kinds of representations achieve comparable performance decreases exponentially as the number of task conditions increases. Consequently, already at modest task complexity involving tens of conditions, the output coding level for which high and low-dimensionality representations have comparable capacity is too low to be physiologically or computationally relevant.

A simple entropic argument shows that uniformly sampling the binary classifications overwhelmingly favors the classifications with output coding level $f = 0.5$, that is classifications whose outputs are roughly evenly split between being 0 and 1. If the output coding level is

constrained to be lower, that is if we only consider classifications with a larger fraction of 0's as output, we will considerably reduce the number of available binary classifications. More precisely, as we prove in this section, the total number of binary classifications with a fixed output coding level f drops exponentially with the squared difference between f and the maximal coding level of 0.5. However, the fraction of implementable classifications among this reduced set can be comparably larger. If the two effects appropriately compensate each other, it might be convenient to reduce the coding level.

Here we show that this can only be the case for very low task complexity involving very few conditions. Already for a number of conditions corresponding to that of our task, the only way to get performances that are comparable to those of high-dimensional representations with pure-selectivity neurons is to go to coding levels that are so low to be physiologically unrealistic and computationally disadvantageous.

Comparison of N_c achieved by recorded high-dimensional representations and low-dimensional pure selectivity representations for sparse outputs

We compute the number of implementable binary classifications with at most 10% cross-validation error $\langle N_c \rangle$ versus the output coding level f. the plot is shown in Fig. S12 for both, recorded neurons and pure selectivity representations. As expected, for very low coding levels both curves decrease and tend to approach each other as f becomes lower. However, if we assume that the coding level of the output is self-consistently equal to the coding level of the input (i.e. the measured PFC coding level; cfr. Fig. S11), we should be interested in what happens at a realistic coding level of $f \approx 0.31$. As we can see from Fig. S12, at coding level $f = 0.31$ (indicated by dotted vertical line) the number of implementable binary classifications remains 2 orders of magnitude larger for the high-dimensional representations of real neurons with respect to pure selectivity representations.

Notice that in the case of the experiment that we analyzed the total number of conditions is 24, and hence it is impossible to go below $f = 1/24 \sim 0.04$ and it is difficult to draw general conclusions for values of f underneath that cutoff. It is anyway clear from the graph in Fig. S12 that N_c for real neurons and pure selectivity neurons only start to converge to each other for very low coding levels where N_c is several orders of magnitudes lower than its maximum at $f = 0.5$.

As we will show in the next paragraph, this is a general trend that becomes even more accentuated as task complexity and task conditions increase.

Theoretical scaling of the number of binary classifications for varying output coding level

Here we compute how the fraction of implementable binary classifications depends on the output coding level.

Given a number of c outputs there exists 2^c binary strings corresponding to the existing binary classifications with c inputs. We now want to restrict ourselves to the binary classifications with a fixed coding level, that is those with a fixed number of ones in the

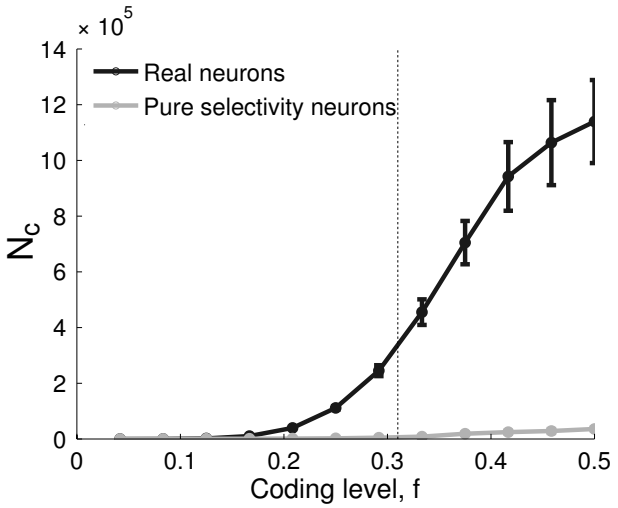

Figure S12: Number of binary classifications that are implementable with at most 10% cross-validation error, as a function of the output coding level f with 2048 neurons in the input. The plot compares this quantity for the real recorded representations and the pure selectivity representations. As expected, for very low coding levels both curves decrease and tend to approach each other. However, for measured coding levels (dotted vertical line indicating $f = 0.31$, cfr. Fig. S11) the number of implementable binary classifications remains 2 orders of magnitude larger for the high-dimensional representations of real neurons with respect to pure selectivity representations.

output. The question is, how many binary classifications with a fixed coding level f exist? The ultrasparse case, is for instance the one in which only one out of c outputs is set to one. This corresponds to $f = 1/c$ and to a number c of binary classifications. In general we are interested in knowing how many classifications out of the total number of possible have a fixed coding level f.

We can indirectly answer this question by computing the ratio between the number of binary function with a coding level f and those with maximal coding level 0.5. This number corresponds to the following ratio between binomial coefficients:

$$\mathrm{F}(c, f) = \binom{c}{f \cdot c} \Big/ \binom{c}{c/2} .$$

It will be convenient to compute the natural logarithm of this number. It will also be convenient to consider deviations of f from 0.5, so we introduce the variable $\Delta = \frac{1}{2} - f$.

Using Stirling's approximation for $\ln n$ in the definition of the binomial coefficient we get

$$\ln \binom{c}{d} = \left(c + \frac{1}{2}\right) \ln c - \left(d + \frac{1}{2}\right) \ln d - \left(c - d + \frac{1}{2}\right) \ln(c - d) - \frac{1}{2} \ln 2\pi + O\left(\frac{1}{c}\right) + O\left(\frac{1}{d}\right),$$

which we can use in the previous expression to get

$$\ln \mathrm{F}(c,f) = \ln \binom{c}{f\cdot c} - \ln \binom{c}{c/2} = -(c+1)\left(\ln\left(\frac{c}{2}-\Delta c\right)+\left(\ln\frac{c}{2}\right)+\Delta c\right) + \Delta c \ln\left(\frac{c}{2}+\Delta c\right) - \Delta c \ln\left(\frac{c}{2}-\Delta c\right) + O\left(\frac{1}{\Delta c}\right).$$

Expanding $\ln\left(\frac{c}{2}\pm\Delta c\right)$ in a Taylor series around $\ln\left(\frac{c}{2}\right)$ we finally get

$$\ln \mathrm{F}(c,f) = -2c\Delta^2 + 2\Delta^2 + O\left(\frac{1}{\Delta c}\right) + O(\Delta^3).$$

For this number to be of order one we have to work with the scaling $\Delta \sim 1/\sqrt{c}$, so we fix $\Delta = \frac{g}{2\sqrt{c}}$ with g of order one and get

$$\ln \mathrm{F}(c,f) = -\frac{g^2}{2} + O\left(\frac{1}{\sqrt{c}}\right),$$

for coding level f that scales in the following way

$$f = \frac{1}{2} - \frac{g}{2\sqrt{c}}.$$

This shows that the fraction of implementable binary function decreases exponentially as the coding level f moves away from $1/2$, and as c increases the factor in the exponent is proportionally higher:

$$\mathrm{F}(c,f) \sim \exp\left(-2c\left(\frac{1}{2}-f\right)^2\right).$$

S.13 Verification that firing rate is not different during correct and error trials

QUESTION: Is the firing rate of the neurons different during correct and error trials?
ANSWER: No, the comparison of the firing rate of the recorded neurons during correct and error trials does not reveal any significant difference across the population of neurons.

We verify that there is no appreciable difference in the firing rate between correct and error trials. The scatter plot in Fig. S13 shows the overall firing rate of every neuron in correct and error trials in the two-object delay epoch, plotting them against each other. Every point in the plot corresponds to a different neuron. Points on the red diagonal line correspond to neurons whose average firing rates are equal in correct and error trials. The figure does not show any large or systematic deviation from the diagonal, compared to what is expected

from Poisson noise (whose average effect is indicated by the gray shading).

More quantitatively, a Wilcoxon signed-rank test does not report any statistically significance difference between the distributions of firing rates (Wilcoxon rank-sum test, $p > 0.9$).

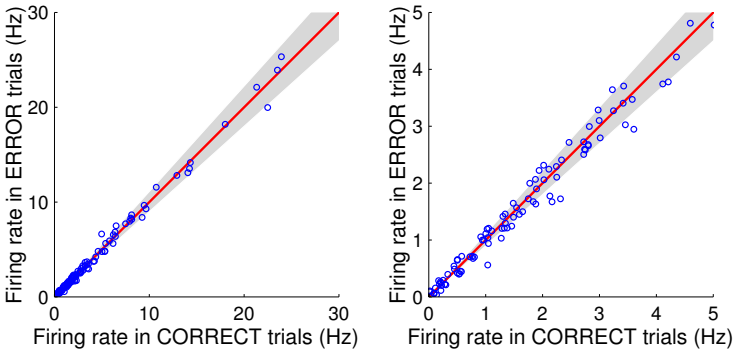

Figure S13: Overall firing rate in error trials as a function of firing rate in correct trials during the two-object delay epoch for the 121 neurons for which we have enough recall task trials to carry out the dimensionality analysis (at least 4 trials per condition). For these neurons we had on average ~ 208 correct trials and ~ 200 error trials per neuron. The panel on the right is a zoom-in on the panel on the left in a range between 0 and 5 Hz. The red diagonal indicates the points where the graphed dots should fall in the case of exactly equal firing rates in correct and error trials. The gray shading indicates the deviations away from the diagonal that can be explained by finite sampling of underlying Poisson spike counts (standard error of the mean).

S.14 Verification that single cell variability is not different during correct and error trials

QUESTION: Is the variability across neurons different during correct and error trials?
ANSWER: No, the comparison of the Fano factor of the recorded neurons during correct and error trials does not reveal any statistically significant difference.

We verify that there is no statistically significant difference between the Fano factor measured in correct and error trials in the two-object delay epoch during the execution of the recall task. For every neuron we compute the Fano factor in every condition. We then average the results across conditions and obtain for every neuron an average Fano factor during correct and error trials. The scatter plot in Fig. S14 shows these average Fano factors, plotting them against each other. Every point in the plot corresponds to a different neuron. Points on the red diagonal line correspond to neurons whose average Fano factors are equal in correct and error trials. The figure does not show any large or systematic deviation from the diagonal, and a Wilcoxon rank-sum test indicates that the distribution does not change significantly between correct and error trials (Wilcoxon rank-sum test, $p > 0.6$).

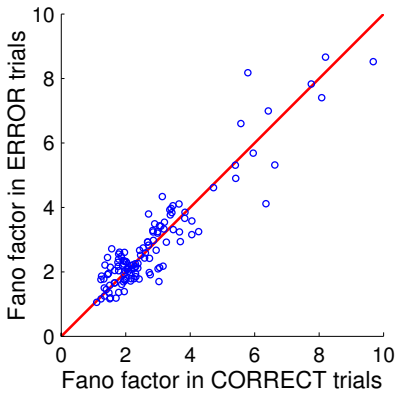

Figure S14: Comparison of the average Fano factor during correct and error trials in the two-object delay epoch of the recall task. Every dot in the graph indicates the Fano factor of a recorded neuron in the two-object delay epoch averaged across conditions computed during correct trials and plotted against the same quantity during error trials. The figure does not show any large or systematic deviation from the diagonal, and a Wilcoxon rank-sum test indicates that the distribution does not change significantly between correct and error trials (Wilcoxon rank-sum test, $p > 0.6$).

S.15 Verification that coding level is not different during correct and error trials

QUESTION: Is the coding level of the neurons different during correct and error trials?
ANSWER: No, the comparison of the coding level of the recorded neurons during correct and error trials does not reveal any statistically significant difference.

We verify that there is no statistically significant difference between the coding level measured in correct and error trials in the two-object delay epoch during the execution of the recall task. We compare the distributions of coding levels obtained as in Supplementary Section S.11 (but only during the two-object delay epoch of the Recall task) with the method developed by [36] (Fig. S15, left panels) and by [37] (Fig. S15, right panels). Fig. S15a shows the distribution of these quantities across neurons during correct trials, while Fig. S15b shows them for error trials.

The median values of the distributions (dashed vertical lines on the plots) between the correct and error trials show a minimal change which is below 5%. In addition to that, a Wilcoxon signed-rank test does not report any statistically significance difference between the distributions (Wilcoxon rank-sum test, $p > 0.7$ for f_{VG} and $p > 0.6$ for $f_{1/2}$).

In both conditions and according to both measures less than 2.5% of the neurons reach a sparseness that goes below 1/12, the level of a grandmother-type cell selective to a unique condition in the recall task.

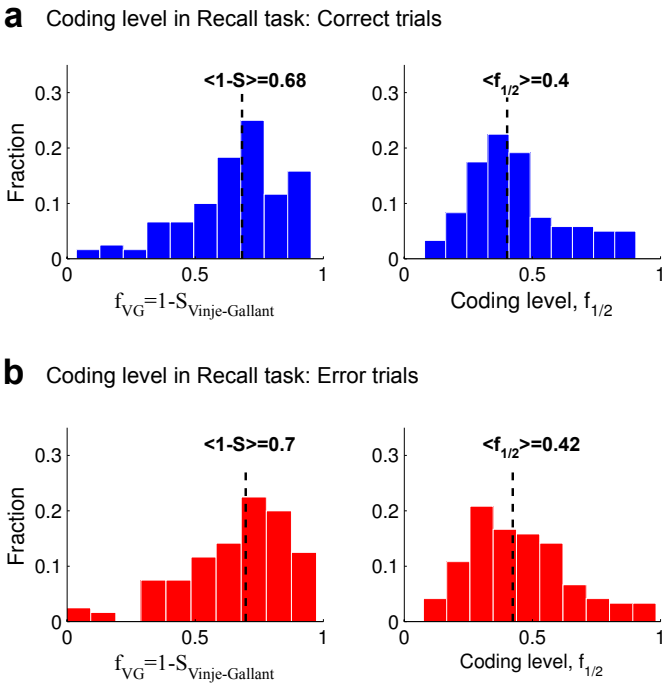

Figure S15: Comparison of coding level during correct and error trials in the two-object delay epoch of the Recall task. **a**, Distribution of coding level as computed in [36] (left panel) and [37] (see Supplementary Section S.11) for correct trials during Recall task. **b**, Same as in **a** but for error trials. The median values of the distributions (dashed vertical lines on the plots) between the correct and error trials show a minimal change which is below 5%. Moreover, the distributions are statistically not significantly different (Wilcoxon rank-sum test, $p > 0.7$ for f_{VG} and $p > 0.6$ for $f_{1/2}$).

S.16 Verification that the decoding of stimuli during error trials is not explained by a ceiling effect

QUESTION: Fig. 5b in the main text indicates that the identity of the visual cues can be decoded equally well during error and correct trials. Is this due to a ceiling effect? Both the performance in the correct and in the error trials saturate at 100%, and it could be that the readout signals are both strong, but significantly different.
ANSWER: We checked that the similarity of the decoding performance of the decoder in correct and error trials is not due to a ceiling effect. We reduced the number of neurons the decoder reads out until the maximal performance drops to values that are far from saturating at 100 %. Even when the performance does not saturate the decoding performance is the same in correct and error trials.

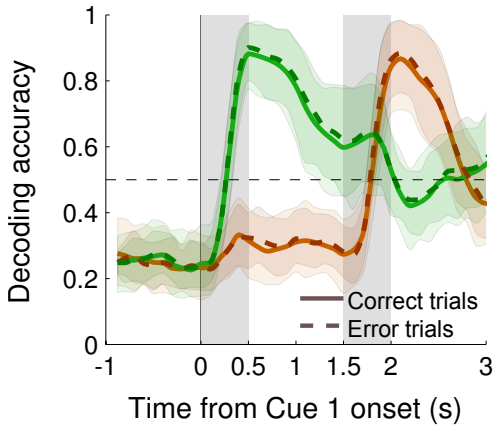

Figure S16: The decoding performance in the correct (continuous lines) and error trials (dashed lines). The identity of the two cues can still be decoded in the error trials, even with as few as 25 neurons. Remarkably, the decoding performance is the same in the correct and error trials. This figure is analogous to Fig. 5b in the main text, except for the fact that the decoder is trained and tested on a limited number of neurons (25), in order to verify that the decoding performance in correct and error trials coincide even when they do not saturate at 100%.

S.17 Principal Component Analysis of the difference between correct and error trials

QUESTION: Fig. 5 shows that the dimensionality of neural representations collapses in the error trials. This decrease can be ascribed to the changes in the non-linear mixed selectivity component of the neuronal responses. The analysis revealing this phenomenology is based on the estimate of dimensionality from the number of implementable classifications. Would the estimates based on principal component analysis be consistent with these results?
ANSWER: Not only the analysis is consistent, but it also reveals other interesting facts: 1) the first 6 PCs seem to account mostly for the pure selectivity component of the neuronal responses, and they remain unchanged in error trials. We show that the first 3 PCs mostly encode the second cue, and the other 3 encode, more weakly, the first cue (as expected, each cue contributes to 3 dimensions). 2) PCs 7,8 seem to account for the dimensionality change.

From Fig. S17 we see that the first 6 principal components (PC) isolated by the PC Analysis cluster in two groups of three PCs. In addition, these two groups of 3 PCs do not vary significantly between correct and error trials. The change in dimensionality between the two conditions is rather due to a decrement in the variance explained by principal components 7 and 8.

Given the structure of the task, we hypothesize that the first three principal components whose eigenvalues are singled out by the PCA analysis (Fig. S17) explain the variance due to varying Cue 2 (the most proximal cue to the time bin in which we carry out the analysis). In fact, because every cue can assume 4 distinct identities, the space described by varying cue 2 spans 3 dimensions. Analogously, the second group of three principal components are

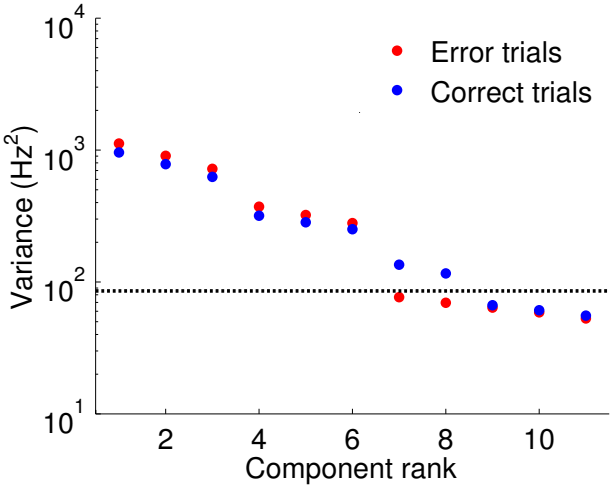

Figure S17: Principal component analysis: variance explained by the principal components in correct (black) and error (red) trials. Graph obtained with 1000 resampled neurons. The dotted line indicates the noise floor computed with all trials (correct and error) as in [15]. Notice that only PCs 7 and 8 change significantly and cross the dashed line in the error trials.

hypothesized to correspond to the variability attributed to Cue 1. If this were the case, the remaining components (from 7 on) represent mixed terms that are due to the variability induced by simultaneously changing the two cues. These would therefore correspond to non-linear mixed selectivity components.

Through a simple correlation analysis we additionally verified that the first 6 PCs indeed correspond to pure selectivity components due to individually varying the inputs. Moreover, the lower PCs in the rank (7,8 and 9) do not show particular sensitivity to individually varying either of the cues.

In conclusion the PCA analysis confirms what we have already determined in Figs 5c,d with our analysis based on the number of implementable classifications. The collapse in dimensionality seems to be due to the non-linear mixed selectivity component of then neuronal response. Indeed, the variance explained by the first 6 PCs, which seem to account for pure selectivity components, remain unchanged in the error trials. This is why cue 1 and cue 2 can still be decoded with high accuracy. The collapse in dimensionality seems to be due to the change in the variance explained by PCs with rank$>$ 6, which presumably contain the non-linear mixed selectivity components, as indicated by Figs 5c,d. Interestingly, what changes in the error trials are PCs that explain a relatively small percentage of the variance and that seem to be crucial for performing the task. This is a warning to consider also PCs that do not seem to be important when one focuses only on the percentage of variance that is explained.

S.18 A simple model for the dimensionality decrease in error trials

QUESTION: In the error trials the dimensionality of the neural representations collapses.

What could be the cause of this collapse?
ANSWER: We speculate that the recorded neurons integrate multiple sources of information. Some of them are relevant for the task (cue 1 and 2 identities and task type), some others are not, but they are a component of the mental state of the animal and they can affect its behavior. We hypothesize that in the error trials the noise due to the task-irrelevant components is significantly larger than in the correct trials. Here we show in a toy model that the neurons that mix non-linearly the input sources are more sensitive than pure selectivity neurons to this form of noise. The dimensionality decreases as a consequence of the reduction in the signal and an increase in noise, two effects that linearize the response of the neurons and preferentially suppress non-linear mixed selectivity. In this scenario, the pure selectivity neurons can still encode individual aspects of the task, as observed in the data.

One of our main results is the observation that in the errors trials the dimensionality of the neural representations collapses. Despite that, we showed that it is still possible to correctly decode the identity of the two visual cues, a seemingly paradoxical observation. In this section we develop a toy model where these two apparently contradictory effects coexist. This scenario comes about because of a form of noise that strongly reduces the non-linear mixed selectivity component of neural activity, causing the dimensionality to collapse, while leaving at the same time pure selectivity responses essentially unaltered, allowing in this way for the decoding of individual task-relevant aspects. This model offers a proof of principle for the reconciliation of the apparent paradox consisting in the decrease in dimensionality during error trials when decoding performance of task-relevant factors remains unaltered.

Our model is based on the idea that, besides pure and non-linear mixed selectivity to the task-relevant variables under experimental control, the neurons' activity is modulated by additional 'uncontrolled variables'. These variables are supposed to complement the description of the subject's mental state, that is the current disposition to behavior [9, 38, 39]. Figs S18a,b illustrates that we model this situation by assuming that the recorded neurons receive inputs from four distinct upstream neural populations: three that represent the task-relevant variables, and one that represents the uncontrolled variables.

In line with the notion that the neural activity of an engaged subject transits through reproducible mental state sequences across different trials (see [39] for a behavioral analysis in a free-operant task that validates this kind of assumption, and see [40] for the evidence of such a mechanism pertaining to conscious perception), our model assumes that in correct trials the patterns of activity encoding the uncontrolled variables can vary among a restricted set of states (Fig. S18a). On the other hand, in the erroneous trials (Fig. S18b), this set is assumed to be significantly larger, i.e. more variable, modeling the idea that the mental state of the animal is in one among numerous possibilities. Notice that this set includes the cases in which the task-relevant variables are still correctly encoded, but they cannot be processed as in the correct trials because of the noise generated by the uncontrolled variables.

Fig. S19 illustrates geometrically the origin of the differential impact of the noise on pure selectivity neurons (Figs S19a,b) and non-linear mixed selectivity neurons (Figs S19c,d). We consider a neuron that receives inputs from three neuronal populations. Two of them

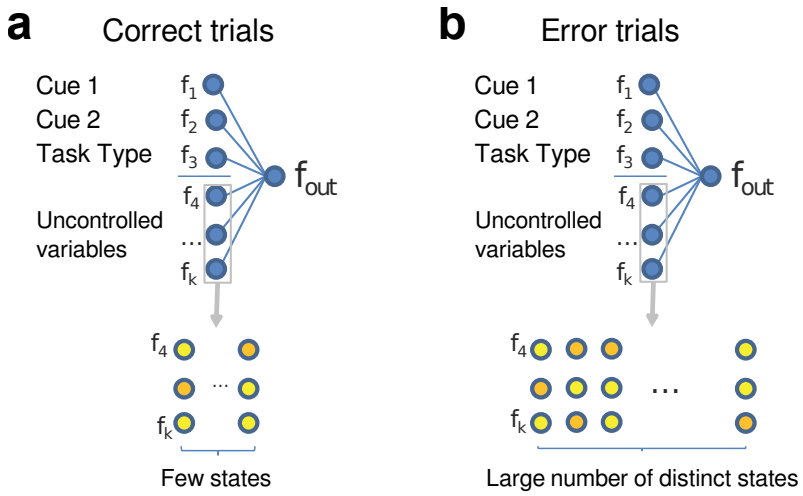

Figure S18: The toy model assumes that the recorded neurons receive inputs from four distinct upstream neural populations: three that represent the task-relevant variables, and one that represents the uncontrolled variables. **a**, In correct trials the patterns of activity encoding the uncontrolled variables can vary among a restricted set of states. **b**, In the error trials the set of possible values for the uncontrolled variables is assumed to be significantly larger, i.e. more variable, modeling the idea that the mental state of the animal is in one among numerous possibilities.

represent task-relevant variables (e.g. cue 1 and cue 2 identity), and the third one encodes the uncontrolled variables. For simplicity we assume that the task-relevant variables have only two values each (e.g. cue 1=A,B; cue 2=C,D). The populations encoding these two variables are homogeneous in the sense that all neurons behave in the same way, and they are either active or inactive. The input space can be represented as in Fig. 1 on a plane, where the x-axis represents the firing rate f_1 of any neuron of the first population and the y-axis the firing rate f_2 of a neuron of the second population (see Fig. S19, left panels). Each point on this plane is one input pattern of activity encoding the two task-relevant variables. These inputs are fed into a model neuron which has activity f_{out}:

$$f_{out} = H(w_1 f_1 + w2 f_2 + \xi - \theta),$$

where $H(\cdot)$ is the Heaviside function (one if the argument is positive, zero otherwise), the w's are the synaptic weights, ξ is the noise generated by the uncontrolled variables and θ is the activation threshold. This model neuron is designed to model the recorded neuronal responses that we analyzed. Depending on the synaptic weights, it may have pure (Figs S19a,b) or non-linear mixed selectivity (Figs S19c,d).

The graphs of Fig. S19 are useful to determine the input-output relation $f_{out}(f_1, f_2)$ and to understand how much this relation is affected by the noise of the uncontrolled variables. The dashed separating line divides all inputs into two groups: the inputs on one side of the line are above the threshold and hence activate the output neuron, whereas the inputs on the other side, generate a synaptic current that is below the threshold. Its orientation

depends on the synaptic weights w_1, w_2, whereas its position depends on both the activation threshold θ and on the input ξ generated by the neurons encoding the uncontrolled variables.

The input-output relation is visualized in the right graphs of Figs S19a-d, where we plot f_{out} for each of the four possible inputs. The pure selectivity neuron in Fig. S19a is selective to the variable encoded by f_2 and not to the variable encoded by f_1, and it is active when f_2 is high, inactive when f_2 is low. The non-linear mixed selectivity neuron reads out both neuronal populations and it responds only to a specific combination of inputs (i.e. when both f_1 and f_2 are high).

In correct trials the neurons encoding the uncontrolled variables are in one among a small set of states. Here for simplicity we assume they are in a particular state and ξ does not vary from trial to trial. In the error trials, these neurons can be in one of many possible states. Their synaptic input ξ causes the dashed line to move stochastically from trial to trial (see Figs S19b,d). Notice that the orientation of the line does not change in the figure, as it depends only on the synaptic weights of the neurons encoding the task-relevant variables, and they are assumed to remain constant.

We plotted in Figs S19b,d the distribution of the displacements of the dashed line (in this simple case we assumed it is a Gaussian). There is now some fraction of trials in which the dashed lines moves across one of the input points. The probability that this happens is given by the colored area under the distribution. When it happens, the response of the output neuron changes. When a population of output neurons is considered, the noise of uncontrolled variables has two disruptive effects on the dimensionality of the neural representations: 1) the output noise increases (note that the noise we are considering is not the only source of the variability in the neuronal response, and other forms of noise may be also present and dominant, making it difficult to detect the increase in output noise), 2) the average $\langle f_{out} \rangle$ across output neurons changes for all the four inputs (right plots in Figs S19b,d) and in particular the distance between the minimal and the maximal responses (signal) decreases, as the four spheres tend to flatten on a plane (linearization).

Although the effects are qualitatively the same for pure and non-linear mixed selectivity neurons, there are significant quantitative differences which are illustrated in Figs S19b,d. For the pure selectivity neuron the probability that the separating dashed line moves beyond one of the inputs is significantly smaller than for non-linear mixed selectivity neurons, as the maximal distance between the dashed line and the spheres is larger. In a realistic situation it is also possible that the width of the distribution is smaller, as highly specialized pure selectivity neurons are likely to receive stronger currents from a smaller subset of inputs. In this case the difference with the mixed selectivity case would be even more prominent.

Summarizing: both in the case of pure and non-linear mixed selectivity neurons, the noise due to the inputs that encode unknown variables reduces the signal and increases the noise of the output. However, in the case of non-linear mixed selectivity neurons this effect is significantly stronger. In this scenario, the dimensionality collapses due to the reduction of the signal-to-noise ratio of non-linear mixed selectivity neurons without compromising the ability to decode individual variables (still encoded in pure selectivity neurons).

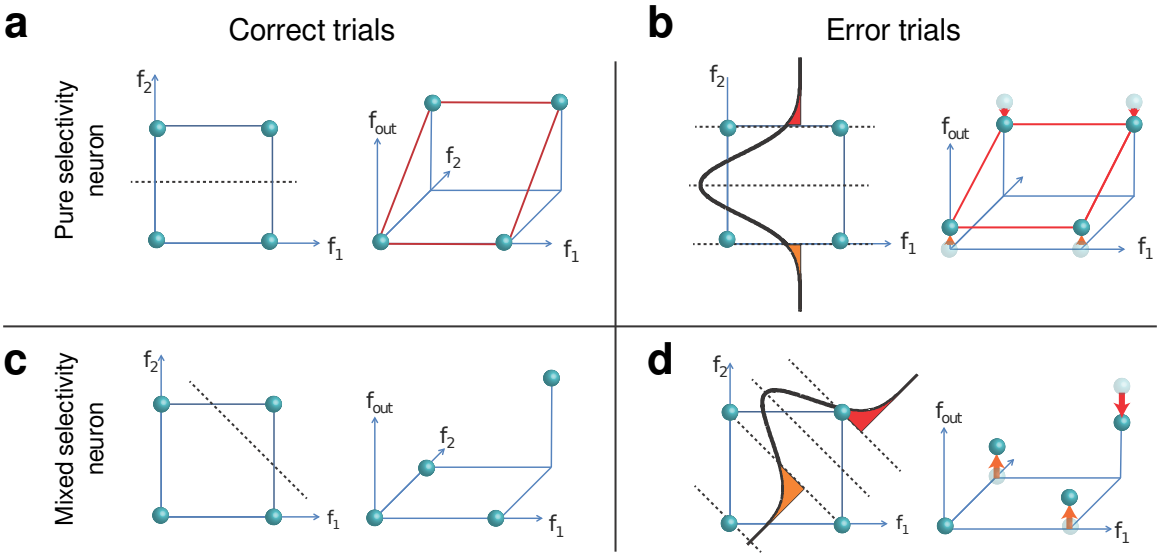

Figure S19: The variability in the mental states leads to a reduction of the signal and an increase in noise. Non-linear mixed selectivity neurons are more sensitive to these changes. **a**,**c**, Left: Diagrams representing the input space for pure/non-linear mixed selectivity neurons in the correct trials. f_1 and f_2 are two inputs representing task-relevant variables. Each point on the plane represents an input pattern of activity. The separating dashed line divides the inputs which are below threshold from those that are above threshold. Assuming that both f_1 and f_2 are either high or low, there are 4 possible inputs, here represented by spheres. **a**,**c**, Right: the input-output relation. f_{out} is the activity of a neuron receiving input from f_1 and f_2. The output is a linear function of the inputs in **a**, for a pure selectivity neuron (see the plane delineated by the red segments), and non-linear in **c**. **b**,**d**, Same as in **a**,**c**, but now in the error trials, in which the inputs encoding the uncontrolled variables generate noise. The orientation of the linear separator depends on the synaptic weights from the two inputs encoding the task-relevant variables. Its position depends on both the activation threshold and on the noisy input generated by the other neurons, those that encode the uncontrolled variables. The position of the separating line is now stochastically distributed across trials (Gaussian curve in the left panel). There is some fraction of trials in which the dashed lines moves across one of the input points. The probability that this happens is the colored area under the distribution. The probability that the inputs are still generating the same output as in the correct trials is an integral over the interval between the colored areas. The diagram on the right represents geometrically the effect of the noise as an average "flattening" of the combined input-output activity patterns. The effects of noise on pure and non-linear mixed selectivity neurons are qualitatively the same, but quantitatively different for pure and non-linear mixed selectivity neurons, as described in the text.

S.19 The contribution of sparse and dense coding neurons to the dimensionality of the neural representations

QUESTION: Grandmother cells are cells that respond only to one condition and hence exhibit very low coding level ($f = 1/24$ in the case of the experiment we analyzed). This type of cell response contains a strong non-linear mixed selectivity component and hence can poten-

tially contribute to the high-dimensionality observed in the recorded neural representations. Can the observed high-dimensionality be ascribed to the contribution of grandmother cells or, more generally, sparse coding cells?
ANSWER: Grandmother cells seem not to be important for the dimensionality for two reasons: 1) most of the recorded cells show a coding level that is significantly above 1/24. 2) the few cells that exhibit sparse coding and may be compatible with grandmother cells coding levels do not greatly contribute to the dimensionality of the representations. Indeed, the removal from the neural representations of the neurons displaying the sparsest levels of activity does not significantly change the dimensionality.

The coding level f of the recorded cells is significantly higher than what one would expect for grandmother cells ($f = 1/24$) at least according to the Vinje-Gallant and Rust-DiCarlo estimations of coding level. In Supplementary Section S.11 we have already analyzed the coding level of the recorded neurons. In Fig. S11 we plotted the distribution of coding levels across neurons, and we show that all of the recorded neurons have a coding level above the grandmother cells level of $1/24$.

We moreover verified that removing the sparsest coding neurons from the dimensionality analysis does not alter the dimensionality. In Fig. S20 we estimate the dimensionality of the neural representation during the two-object delay epoch as in Fig. 4b after removing from the original population 10 percentiles of lowest and highest coding level neurons. The reason why we also removed the neurons with highest coding level is that a neuron with coding level f is post-synaptically equivalent to a neuron with coding level $1 - f$. Removing the 10 lowest and the 10 upper coding level neuron percentiles from the recorded population leaves the cells with coding level above 0.46 and below 0.94 (according to the Vinje-Gallant measure).

Fig. S20 shows that the curve obtained after removing the sparsest cells (green plot) does not significantly differ from the curve obtained with all cells (black curve, taken from Fig. 4b). This confirms that high-dimensionality is not exclusively driven by neurons with extreme coding level.

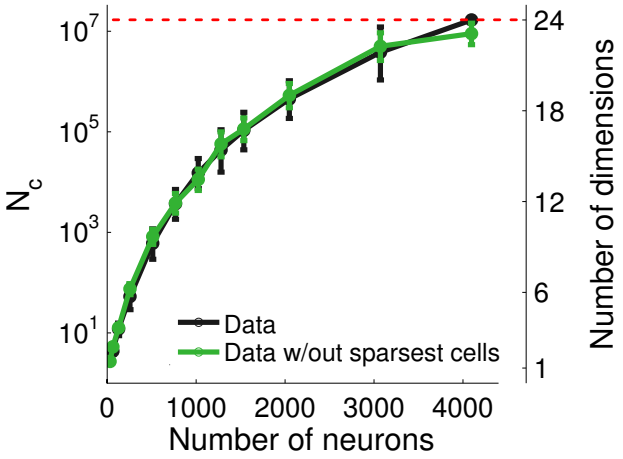

Figure S20: Removing lowest and highest coding level neurons from the recorded population does not significantly change the dimensionality of the neural representations. The plot compares the dimensionality curve for the recorded data shown in Fig. 4b (black curve) with the same curve computed only with a subset of the data obtained by removing cells with Vinje-Gallant coding level below 0.46 and above 0.94, the 10 lowest and 10 upper percentiles (green curve). The two curves do not significantly differ.

S.20 Analysis of test phase epochs in recognition task

QUESTION: How does the dimensionality of the neural representations change in the test phase of the task?
ANSWER: We expect the dimensionality to decrease as soon as some variables become irrelevant for the task. Here we report that we observed the expected drop in the dimensionality.

During the execution of a task the subject has typically to accumulate information about the relevant events. In our case, the animal has first to store the sequence of the sample cues, and then to memorize the test cues. As the number of task-relevant aspects increases, the number of different experimental conditions and, correspondingly, the information to be stored increase combinatorially. A possible way for a neural circuit to efficiently avoid this combinatorial explosion is to discard the information that at some point becomes irrelevant. Correspondingly, we should be able to observe a drop in dimensionality. Notice that this drop is qualitatively different from the collapse observed in error trials. Indeed it reflects the "good design" of the neural circuit which leads to the forgetting of the irrelevant variables, and not a failure of the system. Here we show that, as expected, the dimensionality decreases in the test phase.

In our task the combinatorial increase of the number of conditions in the late epochs of the trial also causes the practical problem of limiting our statistical power, because of the corresponding decrease in the number of recorded trials per neuron per condition. So, for instance, the total number of conditions in the recall task at response time has to be multiplied by all possible configuration of the choice array. This gives $3 \times 2 = 6$ possible

ways to arrange the three objects in the array, times an additional 2 factor, given by the possibilities in choosing the distractor. In total this gives $24 \times 12 = 288$ conditions. The total number of conditions in the recognition task, including the test phase, amounts to $4 \times 3 \times 4 \times 4 = 192$ (test objects are allowed to be repeated). These numbers, with the requirement of having enough trials to cross-validate our results, result in the necessity of recording for sessions of around two-three thousand trials, which is currently not feasible.

However, it is possible to perform a reduced analysis in which some of the conditions are grouped together. Specifically, we examined the recognition task and we considered the conditions that correspond to all combinations of cue 1 and cue 2 identity. For each of these combinations, we considered separately two situations: the test sequences that match the memory sequence, and the test sequences that do not match. The total number of conditions with the additional match/non-match variable is then only $4 \times 3 \times 2 = 24$.

Fig. S21 shows the results of the dimensionality analysis in the 4 delay periods of the recognition task. We observe that during the presentation of the sample cues (Figs S21a,b) the increase in the total number of conditions (from 4 to 12) is accompanied by an increase in the dimensionality of the neural representation. The dimensionality in these intervals is maximal. In the test phase (Figs S21c,d) the dimensionality collapses to levels that are lower than during the sampling phase, despite the maximal dimensionality increases to 24.

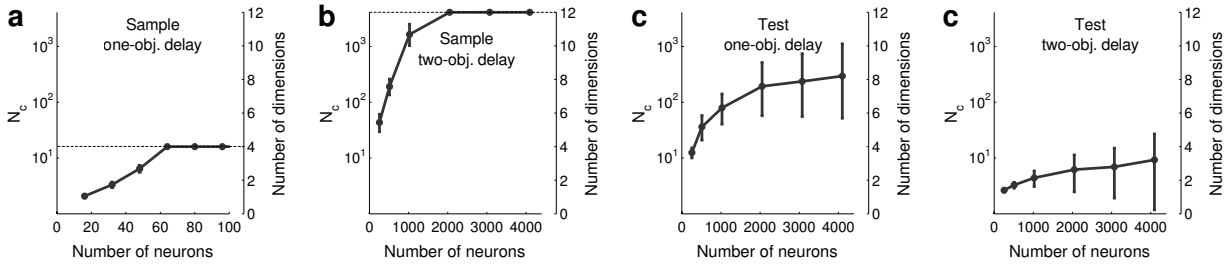

Figure S21: Dimensionality analysis in the recognition task throughout sample and test phases. The conditions are defined with respect to cue 1 and cue 2 identities and to whether the test sequence is a match or non-match. **a**, One-object delay period of sample sequence: dimensionality quickly saturates to the maximum of 4 (the 4 conditions are determined by the 4 possible identities of the cue 1 object). **b**, Two-object delay period of sample sequence: dimensionality quickly saturates to the maximum of 12 (the 12 conditions are determined by the 4×3 possible cue 1 - cue 2 sequences). **c**, One-object delay period of test sequence: dimensionality drops with respect to the previous period. It is much lower than the maximum of 24 (the 12 conditions determined by the 4×3 possible sequences, and the additional 2 possibilities given by whether the first test cue is a match or a non-match). **d**, Two-object delay period of test sequence: dimensionality drops even further with respect to the previous periods. It is much lower than the maximum of 24 (the 12 conditions determined by the 4×3 possible sequences, and the additional 2 possibilities given by whether the test sequence is a match or a non-match).

It is tempting to speculate that this drop in dimensionality might reflect some neural mechanism that actively forgets some of the information that becomes irrelevant for performing the task. Once the animal sees that the first test cue is a non-match, it is no

longer necessary to wait for the second test cue to decide that the whole trial will be a non-match, and the identity of both cues presented during the sample phase can consequently be "forgotten".

This kind of speculations are supported by a population decoding analysis during the test phase, where we see that information about the sample cues strongly decreases in non-match trials compared to match trials (Fig. S22). This decrease may account for the drop in dimensionality observed in Fig. S21.

It is important to notice that it is difficult to interpret what happens in the match trials. Indeed, the test cue is identical to the first cue of the sample sequence, and the ability to decode the first cue may entirely rely on the external drive of the sensory input, and not to the memory of the sample cue.

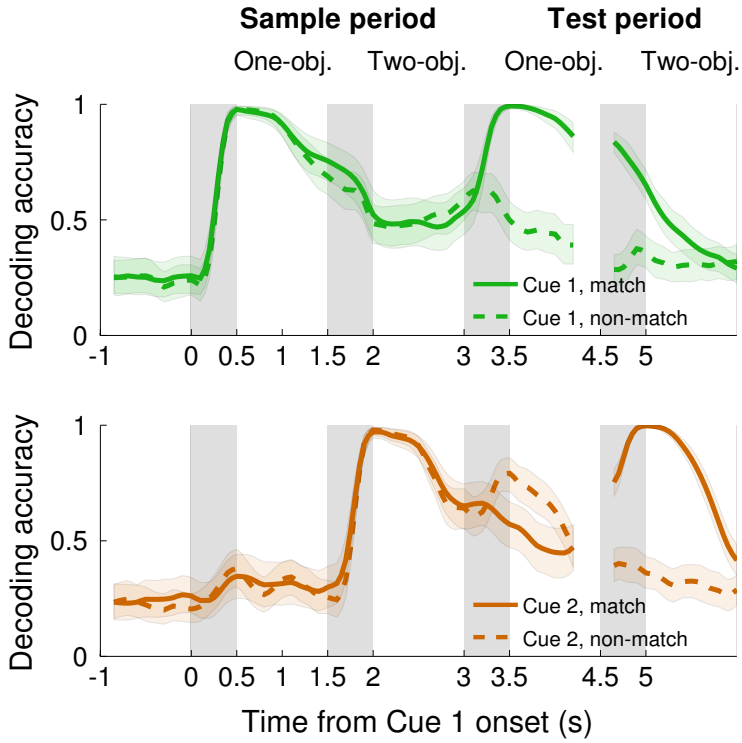

Figure S22: Decoding of cue 1 and cue 2 identity, separately in match and non-match trials of the recognition task. Upper panel: the accuracy throughout the sample and test phases in decoding the identity of cue 1. Before the line breaks trials are sorted according to cue 1 being a match or non-match, after the line breaks they are sorted according to the whole sequence being a match or non-match. Solid line: test cue 1 is a match, and after line break both test cues are a match. Dashed line: cue 1 is a non-match, and after the line break the sequence are not a match. Lower panel: same as in upper panel, but for the decoding of cue 2. Trials are sorted in the same way (for the solid line cue 1 is a match, cue 2 is also a match when it is tested, while for the dashed line cue 1 is a non-match, and when cue 2 is presented the whole sequence is a non-match).

Supplementary References

[24] Dietterich, T. and Bakiri, G. *Journal of Artificial IntelligenceResearch* **2**, 263286 (1995).

[25] Anlauf, J. and Biehl, M. *EPL (Europhysics Letters)* **10**, 687 (1989).

[26] Barak, O. and Rigotti, M. *Neural Comput* **23**(8), 1935–1943 Aug (2011).

[27] Krauth, W. and Mézard, M. *Journal of Physics A: Mathematical and General* **20**(11), L745–L752 (1987).

[28] Breiman, L. *Machine learning* **24**(2), 123–140 (1996).

[29] Hertz, J., Krogh, A., and Palmer, R. G. *Introduction to the theory of neural computation.* (1991).

[30] Minsky, M. and Papert, S. *Perceptrons.* Cambridge: MIT Press, (1969).

[31] Zohary, E., Shadlen, M. N., and Newsome, W. T. *Nature* **370**(6485), 140–143 Jul (1994).

[32] Abbott, L. F. and Dayan, P. *Neural Comput* **11**(1), 91–101 Jan (1999).

[33] Cover, T. *Electronic Computers, IEEE Transactions on* (3), 326–334 (1965).

[34] Grant, D. and Berg, E. *Journal of experimental psychology* **38**(4), 404 (1948).

[35] Pearson, K. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2**(11), 559–572 (1901).

[36] Vinje, W. E. and Gallant, J. L. *J Neurosci* **22**(7), 2904–2915 Apr (2002).

[37] Rust, N. and DiCarlo, J. In *Vision Science Society Annual Meeting*, (2009).

[38] Salzman, C. D. and Fusi, S. *Annu Rev Neurosci* **33**, 173–202 (2010).

[39] Eldar, E., Morris, G., and Niv, Y. *J Neurosci Methods* **201**(1), 251–261 Sep (2011).

[40] Schurger, A., Pereira, F., Treisman, A., and Cohen, J. D. *Science* **327**(5961), 97–99 Jan (2010).