The geometry of abstraction in hippocampus and pre-frontal cortex

Silvia Bernardi*[2,3,8], Marcus K. Benna*[1,4,5], Mattia Rigotti*[7], Jérôme Munuera*[1,9], Stefano Fusi†[1,4,5,6] & C. Daniel Salzman†[1,2,5,6,8]

[1]*Department of Neuroscience, Columbia University*
[2]*Department of Psychiatry, Columbia University*
[3]*Research Foundation for Mental Hygiene*
[4]*Center for Theoretical Neuroscience, Columbia University*
[5]*Mortimer B. Zuckerman Mind Brain Behavior Institute, Columbia University*
[6]*Kavli Institute for Brain Sciences, Columbia University*
[7]*IBM Research AI*
[8]*New York State Psychiatric Institute*
[9]*Current address: Institut du Cerveau et de la Moelle Epinière (UMR 7225), Institut Jean Nicod, Centre National de la Recherche Scientifique (CNRS) UMR 8129, Institut Étude de la Cognition, École normale supérieure*

* These authors contributed equally, † co-senior authors

Abstraction can be defined as a cognitive process that identifies common features - abstract variables, or concepts - shared by many examples. Such conceptual knowledge enables subjects to generalize upon encountering new examples, an ability that supports inferential reasoning and cognitive flexibility. To confer the ability to generalize, the brain must represent variables in a particular 'abstract' format. Here we show how to construct neural representations that encode multiple variables in an abstract format simultaneously, and we characterize their geometry. Neural representations conforming to this geometry were observed in dorsolateral pre-frontal cortex, anterior cingulate cortex and the hippocampus in monkeys performing a serial reversal-learning task. Similar representations are observed in a simulated multi-layer neural network trained with back-propagation. These findings provide a novel framework for characterizing how different brain areas represent abstract variables that are critical for flexible conceptual generalization.

The capacity to conceptualize information underlies most forms of high-level cognitive processing, including categorizing, reasoning, planning, emotion regulation, and decision-making. Each of these processes rely on the ability of the brain to not merely represent information about the details of a particular event (or "example"), but to represent information that describes one or more features that are also shared by many other examples. These features correspond to abstract "cognitive" variables, or concepts. Knowledge of abstract variables (conceptual knowledge) enables one to generalize and immediately draw conclusions about a newly encountered example[1]. Moreover, the capacity to generalize across multiple abstract variables enhances cognitive and emotional flexibility, enabling one to adjust behavior in a more efficient and adaptive manner.

However, a conceptual framework and corresponding data for understanding how the brain simultaneously represents multiple variables in an abstract format - i.e., how the brain can link a single example to multiple concepts simultaneously - has been lacking.

One possibility is that when a population of neurons represents an abstract variable, all information about specific examples is discarded and only the combination of features essential to the abstract variable is retained. For example, the only information retained in an abstract format could be the feature that all the instances belonging to a conceptual set have in common. However, in this case, generalization applied to a new example can only occur with respect to this single encoded abstract variable. The capacity to link a new example to multiple abstract variables simultaneously promotes flexibility, but it requires neural populations to retain multiple pieces of information in an abstract format. To investigate whether and how variables are represented in an abstract format within a neural population, we trained monkeys to perform a serial reversal-learning task. The task involved switching back and forth between two contexts, where contexts were un-cued and defined by sets of stimulus-response-outcome mappings (or contingencies) that differed in each context. Monkeys utilized inference to perform this task efficiently, as demonstrated by the fact that once they experienced that one trial type had changed its contingencies upon a context switch, they changed their behavior for other trial types in the new context.

We sought to determine whether the variables related to this task, especially the variable 'context', were represented by neuronal populations in an abstract format that could support generalization. Neurophysiological recordings were targeted to the hippocampus (HPC) and two parts of the pre-frontal cortex (PFC), the dorsolateral pre-frontal cortex (DLPFC) and anterior cingulate cortex (ACC). The hippocampus has long been implicated in generating episodic associative memories [2–4] that could play a central role in creating and maintaining representations of variables in an abstract format. Indeed, studies in humans have suggested a role for HPC in the process of abstraction [5]. We also recorded from two parts of PFC, ACC and DLPFC, due to their established role in encoding rules and other cognitive information [6–10]. Neural signals representing abstract cognitive variables have been described in PFC [7,9–11], but prior studies have not tested explicitly whether and how multiple variables are represented in an abstract format within a population of neurons.

Neurophysiological data show that multiple task-relevant variables, including context, operant response, and reinforcement outcome, are represented simultaneously in an abstract format in hippocampus, DLPFC, and ACC. This abstract format was revealed by an analysis of the geometry of the representations, which is characterized by the arrangement of the points representing different experimental conditions in the firing rate space of all recorded neurons. In this firing rate space, the parallelism of the coding directions for each abstractly represented variable was significantly enhanced compared to a random unstructured geometry in which abstraction does not occur. The observed geometry enables generalization across conditions within the recorded neural populations in all three brain areas, a signature of abstraction. In a multi-layered neural network trained with back-propagation, a similar geometry of the representations was observed. These results provide a conceptual and mechanistic framework for understanding how the brain can relate a single example to multiple abstract variables simultaneously within a population of neurons.

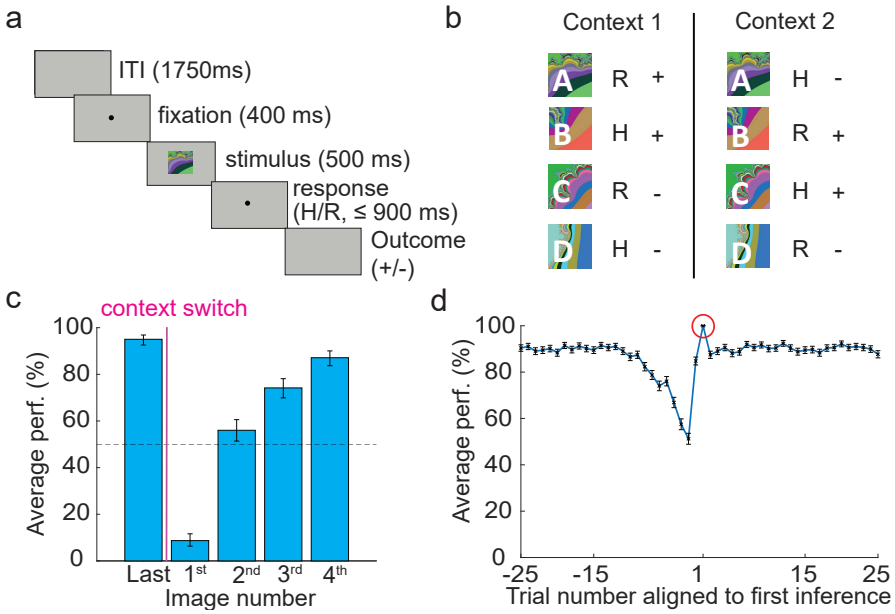


Figure 1: Task design and behavior. a. Sequence of events within a trial. A monkey holds down a press button, then fixates, and then views one of four fractal images (discriminative stimulus). A delay interval follows stimulus viewing during which the operant response (continue to hold or release the press button, respectively H and R, depending on the stimulus displayed) must be performed within 900 ms. After a trace period, a liquid reward is then delivered for correct responses for 2 of the 4 stimuli. For correct responses to the other 2 stimuli no reward is delivered but the monkey is allowed to proceed to the following trial. b. Task scheme, stimulus response outcome map. In each of 2 contexts, 2 stimuli require the animal to continue to hold the button, while the other 2 stimuli require to release the button (H and R). Correct responses result in reward for 2 of the 4 stimuli, and no reward for the other 2 (plus or minus). Operant and reinforcement contingencies are unrelated, so neither operant action is linked to reward per se. After 50-70 trials in one context, monkeys switch to the other context and they continue to switch back-and-forth between contexts many times. c. Monkeys utilize inference to adjust their behavior. Average percent correct is plotted for the first presentation of the last image that appeared before a context switch ("Last") and for the first instance of each image after the context switch (1-4). For image numbers 2-4, monkeys adjusted their behavior to above chance levels despite not having experienced these trials in the current context, thereby demonstrating utilization of inference. Binomial parameter estimate, bars are 95% Clopper-Pearson confidence intervals d. Average percent correct performance plotted as a function of trial number when aligning the data to the first correct trial where the monkey utilized inference (circled in red, see Methods). Performance remains at asymptotic levels once evidence of inference is demonstrated.

Monkeys use inference to adjust their behavior We designed a serial-reversal learning task in which switches in context involve un-cued and simultaneous changes in the operant and reinforcement contingencies of four stimuli. In other words, while the set of stimuli remains constant, two distinct sets of stimulus-response-outcome mappings exist implicitly, one for each context. Correct performance for two of the stimuli in each context requires releasing the button after stimulus disappearance; for the other two stimuli, the correct operant response is to continue to hold the button (Figure 1a,b; see Methods for details). For two of the stimuli, correct performance results in reward delivery; for the other two stimuli, correct performance does not result in reward delivery, but it does prevent both a time-out and monkeys' having to repeat the trial (Figure 1b). Without warning, randomly after 50-70 trials, the operant and reinforcement contingencies switch to the other context; contexts switch many times within an experiment.

On average, the monkeys' performance drops to significantly below chance immediately after a context switch, as the change in contingencies is un-cued (see image number 1 in Fig. 1c). In principle, monkeys could simply re-learn the correct stimulus-response associations for each image independently after every context switch. Behavioral evidence indicates that this is not the case. Instead, the monkeys perform inference. After a context switch, as soon as they have experienced the changed contingencies for one or more stimuli, on average they infer that the contingencies have changed for the stimuli not yet experienced in the new context. Performance is significantly above chance for the stimulus conditions when inference can be applied (see image numbers 2-4 in Fig. 1c). As soon as monkeys exhibited evidence of inference by performing correctly on an image's first appearance after a context switch, the monkeys' performance was sustained at asymptotic levels for the remainder of the trials in that context (Fig. 1d).

The observation that monkeys can perform inference suggests that the different stimulus-response-outcome associations within the same context are somehow linked together. The observed behavior can be explained by utilization of at least two distinct strategies. First, monkeys could utilize a relatively large look-up table to decide what to do. This strategy entails remembering the stimulus-action-outcome of the previous trial and then using this information to select the action in response to the image on the current trial. Since there are 4 possible images that could appear on any given trial, and these images could appear after a trial from either context, the table would contain 32 entries (4 images multiplied by the 8 possible stimulus-action-outcome combinations of the previous trial). The second strategy requires that monkeys create a new abstract variable (a "concept") that pools together all the stimulus-response-outcome combinations (trial types, or instances) that are present within each context to create representations of the two contexts. Upon seeing an image, a monkey can refer to its knowledge of the current context, and then select an action. Both strategies support the inference that we just discussed.

The geometry of neural representations that encode abstract variables We now consider what types of patterns of activity in populations of neurons could support task performance for each of the two strategies that we just described. These patterns could encode information about context in different ways that would correspond to different levels of abstraction. One way to encode the look-up table of the first strategy is to represent each entry of the table with one distinct pattern of activity, which would encode one combination of the current stimulus and one of the 8 possible

stimulus-action-outcome sequences of the previous trial. These 8 sequences could be encoded in the interval preceding the presentation of the stimulus with 8 distinct patterns of activity. When these patterns are random, it is likely that for a sufficient number of neurons the 4 conditions of one context are separable from the 4 conditions of the other context. This is true also in the case in which there is a cloud of points for each of the 8 conditions [12]. Hence, a simple linear decoder can extract the information about context from neural activity in the case of random patterns. Nevertheless, random representations are obviously not in an abstract format, and they would not permit generalization across conditions.

To understand which features of a neural representation enable generalization, the signature of abstraction, it is instructive to consider the geometry of the firing rate space. In this space, each coordinate axis is the firing rate of one neuron, and hence, the total number of axes is as large as the number of recorded neurons.

One simple way to represent context in an abstract format is to retain only the information about context and discard all information about the stimulus, action and reinforcement outcome of the previous trial. Essentially, the neural population would not encode information about specific instances (the details of the prior trial), but would instead encode what is shared between all prior trials of the same context. In this case, the 4 points in the firing rate space that correspond to context 1 would coincide, or, more realistically, in the presence of noise, they would cluster around a single point (see Fig. 2a for an example of simulated data where the activity of 3 neurons are plotted against each other for each of 8 conditions). The other 4 points, for trials occurring in context 2, would form a different cluster. This geometry represents context in an abstract format because it provides a representation that is disassociated from specific instances. Importantly, clustering leads to a geometric arrangement that permits generalization. Indeed, a linear classifier trained to decode context from a small subset of clustered points will generalize right away to all the other points, if the noise is not too large. This is a fundamental property of abstraction that has already been discussed in [5,11].

The clustering geometry we just described allows one to encode only a single abstract variable, but in the real world a single example can often be linked to multiple abstract variables simultaneously. We now show that it is possible to construct neural representations that encode multiple variables in an abstract format at the same time. Consider the simple example that we illustrate in Figure 2b, where we show again the neural representation in the firing rate space in the case of 3 neurons. In this example, the firing rate f_3 of the third neuron in the interval preceding image onset depends only on context and not the stimulus identity, operant action or value of the previous trial. The points of the two contexts lie on two parallel planes that are orthogonal to the 3rd axis. The other two neurons encode the other task-relevant variables as strongly as the third neuron encodes context. In this geometric format, there is no need for clustering, and indeed, the distances between data points within a context are as large as the distances for data points across contexts. However, an abstract representation of context is clearly embedded in the geometry because the third neuron encodes only context and throws out all other information, enabling generalization.

The simple example of Fig. 2b relies on neurons that encode only one variable, context, to provide a representation in an abstract format. Neurons with such pure selectivity are rarely observed, either in our dataset (see below), or more generally in many studies that have demonstrated

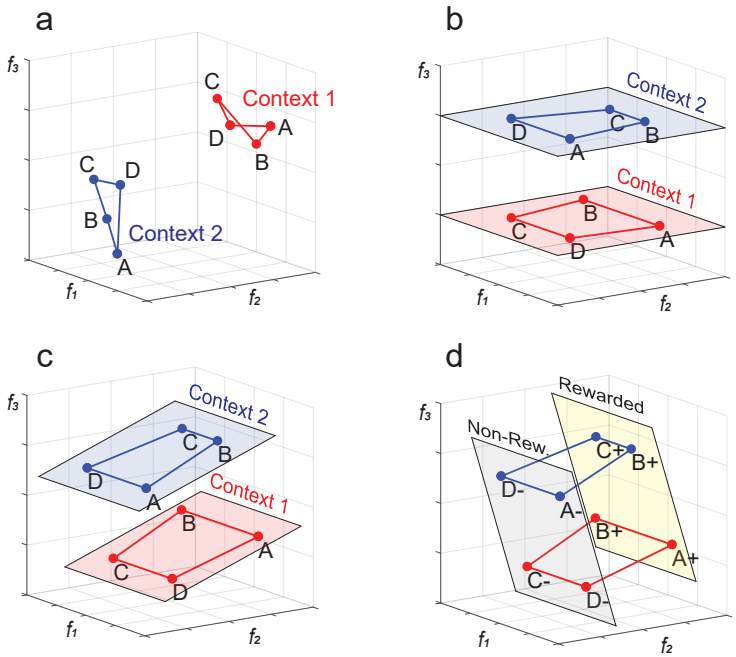


Figure 2: A neural code for abstract variables. a. Abstraction by clustering. Schematic of the firing rate space of three neurons. The mean firing rates corresponding to the eight experimental conditions are clustered according to context. The two contexts are indicated by the colors red and blue, while the eight conditions are labeled by the stimulus identity (A,B,C or D). Due to the clustering, the average within-context distance is less than the mean distance between-contexts. b. Schematic of the firing rate space of three neurons, of which one is specialized for encoding context (f_3 axis). The other neurons encode different variables. The points of each context are in one of the two low-dimensional manifolds (planes in this case) that are parallel. Neurons that are highly specialized to encode only context are rarely observed in the data (see Suppl. Info. S2). c. The same neural representation geometry as in panel b, but rotated in firing rate space, leads to linear mixed selectivity. Even though there are no longer any neurons specialized to encode only context, in terms of decoding as well as generalization using linear classifiers, this case is equivalent to that shown in panel b. d. The same neural geometry as in panel c, but with planes depicted that highlight the encoding of reward value (plus or minus). Data points corresponding to the same value fall within a plane, just as data points corresponding to the same context fall within a plane in the previous panel.

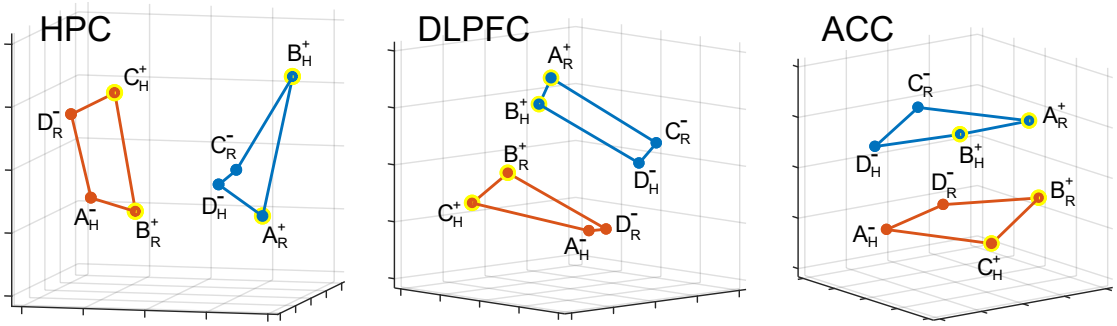


Figure 3: The geometry of the recorded neural representations. Multi-dimensional scaling plots (using Euclidean distances on z-scored spike count data in the 900 ms time window that starts 800ms before stimulus onset) showing the dimensionality-reduced firing rates for different experimental conditions in the three brain areas we recorded from: HPC, DLPFC and ACC. The labels refer to value $(+/-)$ and operant action (R/H) corresponding to the previous trial. Yellow rings are rewarded conditions. While there is a fairly clean separation between the two context sub-spaces, other variables are encoded as well and the representations are not strongly clustered. Note that the context sub-spaces appear to be approximately two-dimensional (i.e., of lower dimensionality than expected for four points in random positions).

that neurons more commonly exhibit mixed selectivity for multiple variables [13,14]. However, all the generalization properties of the representation of context shown in Figure 2b are preserved even when we rotate the points in the firing rate space (see Figure 2c). Here context is still an abstract variable, even though all neurons may now respond to multiple task-relevant variables. Notice that the neural representation constructed in Figure 2c already encodes multiple abstract variables at the same time. Indeed all 3 neurons respond to more than one task-relevant variable, and, more specifically, they exhibit linear mixed selectivity [13,14] to context, operant action and reward value of the previous trial. Context, action and value are also abstract, as illustrated in Figure 2d, where we show the exact same geometry of Figure 2c, but highlight how it encodes also the value of the previous trial in an abstract format. All the points corresponding to the rewarded conditions are contained in the yellow plane, and the non-rewarded points are in the gray plane. These two planes are parallel to each other, just like the ones for the different contexts in Figure 2c. Using a similar construction, it is possible to represent as many abstract variables as the number of neurons. However, additional limitations would arise from the amount of noise that might corrupt these representations.

The geometry of the recorded representations We now show that the geometry of neural representations recorded from monkeys performing the serial reversal learning task conforms to a geometry that supports abstraction, just as we proposed in Figure 2c,d. We will first show that the information about context and other task-relevant variables are present in the recorded neural representations. Then, we will show that this information is encoded in a way that allows for generalization and conforms to the geometry just described.

We recorded the activity of 1378 individual neurons in the PFC and hippocampus in two

monkeys while they performed the serial reversal learning task. Of these, 629 cells were recorded in HPC (407 and 222 from each of the two monkeys, respectively), 335 cells were recorded in ACC (238 and 97 from each of the two monkeys), and 414 cells were recorded in DLFPC (226 and 188 from the two monkeys). Individual neurons exhibit mixed selectivity to all the task relevant variables and their responses are very diverse in all the three brain areas (see Figure S1). The task-relevant variables context, reinforcement outcome (value) and operant action of the previous trial were represented in each of the brain areas, as each variable could be decoded from the firing rates of populations of neurons in each area (Fig. S2a; see Methods). In particular, they could all be decoded in the time interval preceding the stimulus presentation (Fig. S2b,c right panels).

To understand in what format the information about these variables is represented, we used dimensionality reduction methods (multi-dimensional scaling, MDS) to visualize the recorded representations in 3 dimensions (see Methods). This method revealed that the representations appear similar to those illustrated in Figure 2c,d. Figure 3 depicts the MDS plots for all three brain areas, using the same notation as in Figure 2a-d. In HPC, the red and the blue points, which represent the two contexts, are well separated, suggesting some degree of clustering. However, it is clear that also in this case the intra-context distances are not negligible and that the points within the clusters are nicely organized. For example the rewarded and non-rewarded conditions are well separated; this organization is particularly evident in the movies in the Supplementary Material in which these plots can be viewed from many different angles. This indicates that value could also be represented in an abstract format. This type of geometric structure is even more prominent in the DLPFC and ACC. Moreover, the movies in the Supplementary Material suggest that the four points of each context are contained in a low-dimensional subspace, almost a plane, as in the geometry proposed in Figure 2c,d. The planes corresponding to the two contexts are approximately parallel.

Measuring abstraction in high-dimensional spaces: cross-condition generalization and the parallelism score We now introduce two complimentary methods for characterizing the geometry of the neural representations in the original high-dimensional firing rate space. The first method is related to the ability of a linear readout to generalize for multiple variables simultaneously, which is a fundamental property of representing a variable in an abstract format. To illustrate this method, consider for simplicity only a subset of four of the eight conditions in our experiments, where two trial types come from each of contexts 1 and 2, and only one of the trial types in each context is rewarded. A decoder can be trained to classify context only on the conditions in which the monkey received a reward in the previous trial. Thanks to the arrangement of the four points (the lines going through the two points of each context are almost parallel), the resulting decoding hyperplane (gray line in Figure 4a) successfully classifies context when testing on the other two (held-out) conditions in which the monkey did not receive reward. This decoding performance on trials that the classifier was not trained on corresponds to generalization, and it is a signature of the process of abstraction. In other words, if a decoder is trained on a subset of conditions, and it immediately generalizes to other conditions unseen before, without any need for retraining, we conclude that a variable is represented in an abstract format (one that enables generalization). To determine whether the data exhibits the geometry of Figure 2c, we can directly test the ability to generalize by following the same procedure illustrated in Figure 4a: we train a decoder on a subset of condi-

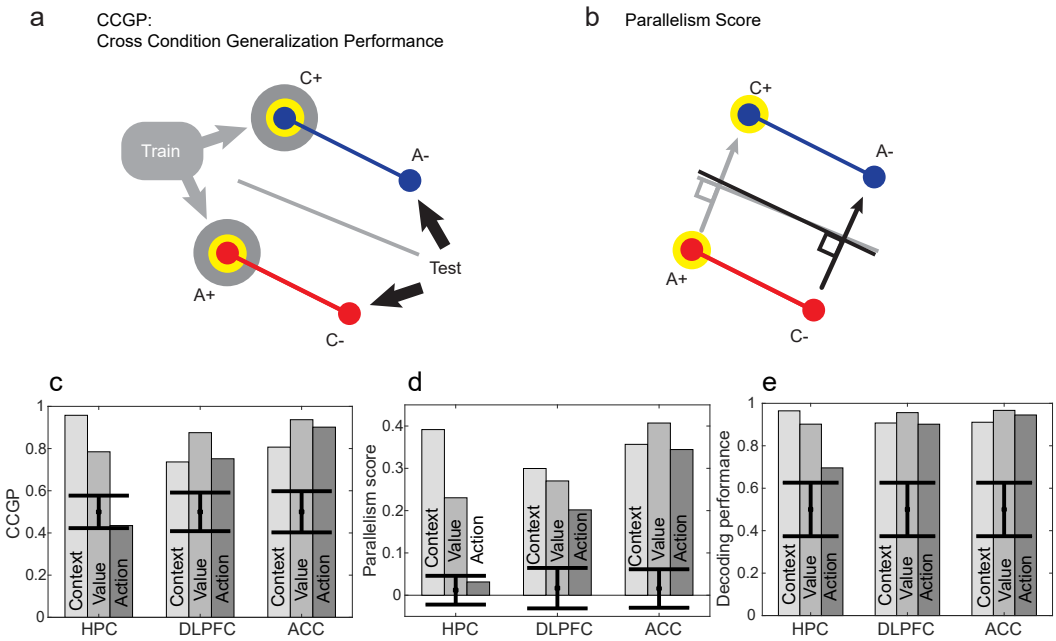


Figure 4: Characterizing abstract geometries in high-dimensional spaces. a. Schematic explanation of cross-condition generalization (CCG) in the simple case of only four experimental conditions, labeled according to context (red versus blue) and value of the stimulus (a yellow ring indicates a rewarded condition). We can train a linear classifier to discriminate context on only two of the conditions (one from each context, in the case shown the rewarded conditions), and then test its generalization performance on the remaining conditions not used for training (here the unrewarded conditions). The resulting test performance will depend on the choice of training conditions, and we refer to its average over all possible (in this case four) ways of choosing them as the cross-condition generalization performance (CCGP). A CCGP that is above chance level would indicate that the neural representation of a given variable is in an abstract format because it enables generalization. b. Schematic explanation of the parallelism score (PS). Training a linear classifier on the two rewarded conditions leads to the gray separating hyperplane which is defined by a weight vector orthogonal to it. Similarly, training on the unrewarded conditions leads to the black hyperplane and weight vector. If these two weight vectors are close to parallel, the corresponding classifiers are more likely to generalize to the other conditions not used for training. In the case of isotropic noise around the two training conditions, these weight vectors will be proportional to the (context) coding vectors connecting the mean neural activities of the training conditions across the context divide. Therefore, instead of training classifiers we can look directly at the angle between these coding vectors, and define a parallelism score as the cosine of the angle between the coding vectors (maximized over all possible ways of pairing up the conditions; see Methods for details and generalization to eight conditions). c-d. CCGP and PS for the context, value and action of the previous trial measured in three brain areas. Context and value are abstractly represented in all three brain areas, but action is abstract only in PFC. e. Decoding performance of the three variables using maximum margin linear classifiers on the average firing rates in the same time interval. Even though the operant action is not abstractly represented in the hippocampus, it can still be decoded with a cross-validated performance significantly above chance level by a simple linear classifier (see also Figure S7 to understand the arrangement of the points in the firing rate space). All data presented were obtained from correct trials in a -800ms to 100ms time window relative to stimulus onset. Error bars are $\pm$ two standard deviations around chance level as obtained from a geometric random model (panel c) or from a shuffle of the data (panels d and e).

tions and test it on the other conditions. We define the performance of the decoder on these other conditions as the cross-condition generalization performance (CCGP).

It is important to stress that the ability to decode a variable like context when training a decoder on a subset of all the trials does not necessarily imply that the CCGP for context is large. For example, if the points corresponding to different conditions are at random positions in the firing rate space, the decoding accuracy can be arbitrarily high if the noise is small, but the CCGP will be at chance level. In other words, in the example of Figure 4a, if the points are at random positions and not arranged as in the figure, the probability that the two test conditions will be on the correct sides of the hyperplane is 0.5, and hence the CCGP would not be different from chance. This is true also even when all four points are very well separated (spanning a 3-dimensional subspace), and context is therefore decodable (for points at random positions, the probability that the points of the two contexts are not separable goes to zero as the number of neurons increases).

The second method we used to characterize the geometry of neural representations posits that there is a specific aspect of the geometry that may account for generalization performance: the degree to which the coding directions determined when training a decoder are parallel for different sets of training conditions. Consider the case depicted in Figure 4b. Here we draw the two hyperplanes (which are lines in this case) obtained when a decoder is trained on the two points on the left (the rewarded conditions, gray) or on the two points on the right (unrewarded conditions, black). The two lines representing the hyperplanes are almost parallel, indicating that this geometry will allow good generalization regardless of which pair of points we train on.

We estimate the extent to which these hyperplanes are aligned by examining the coding directions (the arrows in the figure) that are orthogonal to the hyperplanes. For good generalization, these coding directions should be as close to parallel as possible. This is the main idea behind the parallelism score (PS), a measure described in detail in the Methods. A large PS indicates a geometry likely to permit generalization and therefore the corresponding variable would be represented in an abstract format. When multiple abstract variables are simultaneously represented, the PS should be large for all variables, constraining the points to approximately define a geometry of the type described in Figure 2a,b. As the PS focuses on parallelism between coding directions, it can detect the existence of abstract variables even when the neurons are not specialized, or, in other words, when the coding directions are not parallel to the coordinate axes.

In Figure 4 we report both the CCGP and PS measured in the three brain areas during the 900 ms time interval that starts 800 ms before the presentation of the visual stimulus (see Methods). The CCGP analysis reveals that context is abstract in all three areas, and the level of abstraction is comparable across brain areas. An analysis similar to what has been proposed in [5], which only considered clustering for representing an abstract variable, would lead to the wrong conclusion that context is strongly abstract only in HPC (Figure S3). In fact, all three variables are represented in an abstract format in all three brain areas, except the action of the previous trial in the hippocampus. Interestingly, the action can be decoded in HPC (see Figure 4c), even if it is not abstract. Remarkably, the PS exhibits a pattern very similar to the CCGP, indicating a direct correspondence between the geometry of representations and generalization. In conclusion, these analyses show that multiple abstract variables are encoded in the populations of neurons recorded from each studied brain area. Moreover, the geometry of these representations is similar to the one

constructed in Fig. 2c,d.

Abstraction in multi-layer neural networks trained with back-propagation The geometric features of neural representations that we have just described may constitute a general principle that neural networks exhibit when performing cognitive tasks relying on conceptual information. To examine this possibility, we trained a simple neural network model with back-propagation, and asked whether neural representations in the network exhibit similar geometric features as observed in the experiments. Using back-propagation we trained a two layer network (see Figure 5a) to read an input representing a handwritten digit between 1 and 8 (from the MNIST dataset) and to output whether the input digit is odd or even, and, at the same time, whether the input digit is large (> 4) or small (≤ 4) (Figure 5b). We wanted to test whether the learning process would lead to abstract representations of two abstract variables, or concepts: parity and magnitude (i.e. large or small). This abstraction process is similar to the one studied in the experiment in the sense that it involves aggregating together inputs that are visually very dissimilar (e.g. the digits '1' and '3', or '2' and '4'). Analogously, in the experiment very different sequences of events (visual stimulus, operant action and value) are grouped together into what we defined as contexts.

After training the network, we presented input samples that were held out from training, and we 'recorded' the activity of the two hidden layers. The multidimensional scaling plots, similar to those of Figure 3 for the experimental data (but reduced to two dimensions), are shown in Figure 5c for the input layer and for the two hidden layers of the simulated network. Each digit in these plots represents a different input. They are colored according to the parity/magnitude tasks illustrated in Figure 5b. While it is difficult to detect any structure in the input layer (the slight bias towards red on the left side is mostly due to the similarity between '1's and '7's), in the second hidden layer we observe the type of geometry that would be predicted for a neural representation that encodes two abstract variables, namely parity (even digits on the left, odd digits on the right), and magnitude (large at the top, small at the bottom). The digits tend to cluster at the four vertices of a square, which is the expected arrangement.

Just as in the experiments, we computed both the CCGP and the PS. We analyzed these two quantities for all possible dichotomies of the eight digits, not just for the dichotomies corresponding to magnitude and parity. This corresponds to all possible ways of dividing the digits into two equal size groups. In Figure 5d,e we ranked these dichotomies according to their CCGP and their PS, respectively. The largest CCGP and PS correspond to the parity dichotomy, and the second largest values correspond to the magnitude dichotomy (circles marked by crosses in Figure 5d,e). For these two dichotomies, both the CCGP and the PS are significantly different from those of the random models. There are other PS values that are significant. However, they correspond to dichotomies whose labels denote one or both of the two trained dichotomies. If one restricts the analysis to the dichotomies orthogonal to both of them (filled circles in Figure 5d,e), none are significantly above chance level. This analysis shows that the geometry of the neural representations in the simulated network is similar to that observed in the experiment. Furthermore, the CCGP and the PS can identify the dichotomies that correspond to abstract variables even when one has no prior knowledge about these variables. Indeed, it is sufficient to compute the CCGP and the PS for all possible dichotomies to discover that parity and magnitude are the abstract variables in these

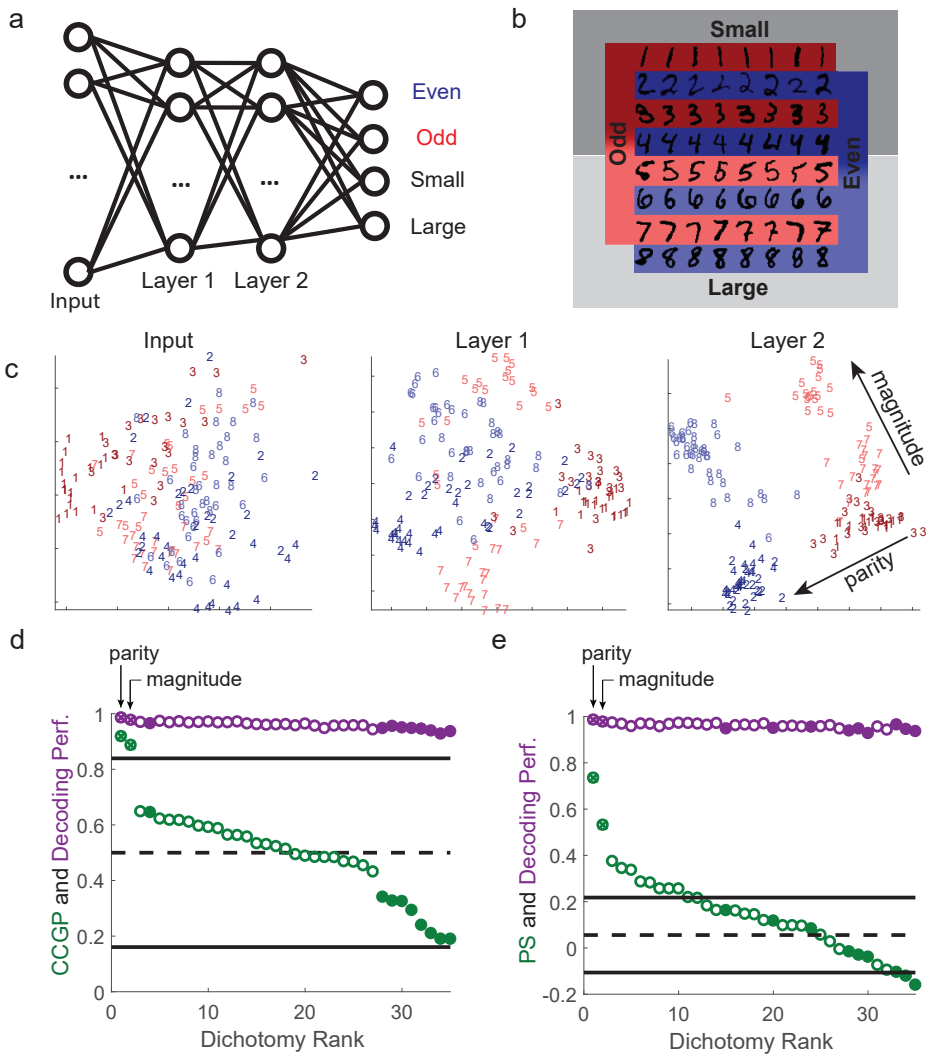
simulations.

Next, we asked whether a neural network trained to perform a simulated version of our experimental task would reveal similar principles. Because of the sequential character of the task, it was natural to model the monkeys' interaction with their environment in the task within a Reinforcement Learning framework [15]. Specifically, we used Deep Q-learning, a technique that uses a deep neural network representation of the state-action value function of an agent trained with a combination of temporal-difference learning and back-propagation refined and popularized by [16]. The use of a neural network is ideally suited for a comparative study between the neural representations of the model and the recorded neural data. As commonly observed in Deep Q-learning, the neural representations that we obtained with Deep Q-learning display significant variability across runs of the learning procedure. However, in a considerable fraction of runs, the neural representations recapitulate the main geometric features that we observed in the experiment S3. In particular, after learning, context is represented as an abstract variable in the last layer, despite not being explicitly represented in the input, nor in the output. Moreover, the neural representations of the simulated network encode multiple abstract variables simultaneously, as observed in the experiment.

Discussion The cognitive process that finds a common feature - an abstract variable - shared by a number of examples or instances is called abstraction. Abstraction enables one to utilize inference and deduce the value of an abstract variable when encountering a new example. Here we proposed a geometric construction of neural representations that accommodate the simultaneous encoding

---

Figure 5 *(following page)*: Simulations of a multi-layer neural network reveal that the geometry of the observed neural representations can be obtained with a simple model. a. Diagram of the network architecture. The input layer receives gray-scale images of MNIST handwritten digits with 784 pixels. The two hidden layers have 100 units each, and in the final layer there are two pairs of output units corresponding to two binary variables represented by a concatenation of two one-hot vectors. b. Schematic of the two discrimination tasks. The network is trained using back-propagation to simultaneously classify inputs (we only use the images of digits 1-8) according to whether they depict even/odd and large/small digits. The colors indicate the parity and the shading the magnitude of the digits (darker for smaller ones). c. Two-dimensional MDS plots of the representations of a subset of images in the input (pixel) space, as well as in the first and second hidden layers. While in the input layer there is no structure apart from the accidental similarities between the pixel images of certain digits (e.g. ones and sevens), in the first and even more so in the second layer a clear separation between digits of different parities and magnitudes emerges in a geometry with consistent and approximately orthogonal coding directions for the two variables, which suggests a simultaneously abstract representation for both variables. d. Cross-condition generalization performance (CCGP, green) for the variables corresponding to all possible balanced dichotomies when the second hidden layer is read out. The dichotomies are ranked according to the strength of their CCGP. Only the two dichotomies corresponding to parity and magnitude are significantly different from a geometric random model (chance level is 0.5 and the two solid black lines indicate plus/minus two standard deviations). The decoding performance (purple) is high for all dichotomies, and hence inadequate to identify the variables stored in an abstract format. e. Same as panel d, but for the parallelism score (PS), with error bars obtained from a shuffle of the data. Both the CCGP and PS allow us to identify the abstract variables actually used to train the network.


a
Even
Odd
Small
Large
Input
Layer 1
Layer 2
b
Small
Odd
Even
Large
c
Input
Layer 1
Layer 2
magnitude
parity
d
parity
magnitude
CCGP and Decoding Perf.
Dichotomy Rank
e
parity
magnitude
PS and Decoding Perf.
Dichotomy Rank

of multiple abstract variables. These representations are characterized by a high parallelism score, a measure based on the angles between coding directions for any given variable. Crucially, this geometrical property is related to a useful statistical property - cross-condition generalization performance - that characterizes how linear readouts can generalize across conditions, which is a signature of representing a variable in an abstract format.

We used the parallelism score and cross-condition generalization to characterize the neural activity patterns recorded in a serial reversal learning experiment in which monkeys switched back and forth between two contexts. These measures revealed that the representation of the task-relevant variable "context" was in an abstract format in HPC, DLPFC and ACC prior to the stimulus presentation on a trial. This information can support performance of the task as knowledge of the stimulus identity combined with knowledge of context can be used to select which operant action to perform and to anticipate accurately whether to expect a reward.

In addition to context, all the recorded brain areas represented at least one other variable in an abstract format (the action and reward value of the previous trial in DLPFC and ACC, and the value of the previous trial in HPC). Representations that arise in simple neural network models trained with back-propagation also represent multiple variables in an abstract format simultaneously, as the representations exhibit the same geometric properties that we observed in the experimental data. The observed geometry may therefore be a general feature underlying the encoding of abstract variables in biological and artificial neural systems.

Although information about the value and action of the previous trial are represented in all three brain areas, this information is not needed to perform correctly on the next trial if the animal did not make a mistake. However, at the context switch, the reward received on the previous trial is the only feedback from the external world that indicates that the context has changed. Therefore, reward value is essential when adjustments in behavior are required, suggesting that storing the recently received reward could be beneficial. Consistent with this notion, in the Supplementary Material S3 we show in simulations that value becomes progressively more abstract as the frequency of context switches increases (see Figure S14). In addition, monkeys occasionally make mistakes that are not due to a context change. To discriminate between these occasional errors and those due to a context change, information about value is not sufficient and information about the previously performed action could be essential for deciding correctly the operant response on the next trial. Conceivably, the abstract representations of reward and action may also afford the animal more flexibility in learning and performing other tasks. Previous work has shown that information about recent events is represented whether it is task-relevant or not (see e.g. [17,18]). This storage of information (a memory trace) may even degrade performance on a working memory task [19], but presumably the memory trace might be beneficial in other scenarios that demand cognitive flexibility in a range of real-world situations.

Our analysis showed that DLPFC and ACC represent more variables in an abstract format than hippocampus, as the action of the previous trial is in an abstract format only in DLPFC and ACC. This may reflect the prominent role of pre-frontal areas in supporting working memory (see e.g. [20–22]). Moreover, the fact that the hippocampus represents fewer variables in an abstract format as characterized by the parallelism score and cross-condition generalization explains why if one only considers clustering as a signature of abstraction, context is strongly identified as being in

an abstract format only in the hippocampus (see Figure S3). However, the geometric analysis of patterns of neural activity reveals that pre-frontal cortex also represents context in abstract format. In general, detecting abstract variables becomes more difficult as their number grows, since this increases the dimensionality of the sub-spaces encoding different values of each abstract variable. In this case, one therefore requires more samples in order to generalize, which affects the statistics of the cross-condition generalization performance.

Context, action and value of the previous trial can all be represented in an abstract format in the recorded areas, but context is particularly interesting because it is not explicitly represented in the sensory input, nor in the motor response, and hence it requires a process of abstraction (learning) based on the temporal statistics of sequences of stimulus-response-outcome associations. However, it is important to stress that learning may also be required for creating abstract representations of more concrete variables, such as action, which corresponds to a recent motor response, or value, which encodes a sensory experience, namely recent reward delivery.

Dimensionality and abstraction in neural representations Dimensionality reduction is widely employed in many machine learning applications and data analyses because, as we have seen, it leads to better generalization. In our theoretical framework, we constructed representations of abstract variables that are indeed relatively low-dimensional, as the individual neurons exhibit linear mixed selectivity [13,14]. In fact, these constructed representations have a dimensionality that is equal to the number of abstract variables that are simultaneously encoded. Consistent with this, the neural representations recorded in the time interval preceding the presentation of the stimulus are relatively low-dimensional, as expected (Supplementary S4).

Previous studies showed that the dimensionality of neural representations can be maximal (monkey PFC, [13]), very high (rodent visual cortex[23]), or as high as it can be given the structure of the task [24]. These results seem to be inconsistent with what we report in this article. However, dimensionality is not a static property of neural representations; in different epochs of a trial, dimensionality can vary significantly (see e.g. [25]). Dimensionality has been observed to be maximal in a time interval in which all the task-relevant variables had to be mixed non-linearly to support task performance [13]. Here we analyzed a time interval in which the variables that are encoded do not need to be mixed. In this time interval, the most relevant variable is context, and encoding it in an abstract format can enhance flexibility and support inference. However, during the presentation of the stimulus, the dimensionality of the neural representations increases significantly (Supplementary S4), indicating that the context and the current stimulus are mixed non-linearly later in the trial, similar to prior observations [13,14,23,26,27].

Finally, we should emphasize that our data are also consistent with intermediate regimes in which the coding directions are not perfectly parallel. Distortions of the idealized geometry can significantly increase dimensionality, providing representations that preserve some ability to generalize, but at the same time support operations requiring higher dimensional representations (see Supplementary Information S17).

Other forms of abstraction in the computational literature Machine learning and in particular computational linguistics have recently started to demonstrate impressive results on difficult lexical semantic tasks thanks to the use of word embeddings, which are vector representations whose geometric properties reflect the meaning of the linguistic tokens they represent. One of the most

intriguing properties of recent forms of word embeddings is that they exhibit linear compositionality that makes the solution of analogy relationships possible via linear algebra [28,29]. A well-known example is provided by shallow neural network models that are trained in an unsupervised way on a large corpus of documents and end up organizing the vector representations of common words such that the difference of the vectors representing 'king' and 'queen' is the same as the difference of the vectors for 'man' and 'woman'[28]. These word embeddings, which can be translated along parallel directions to consistently change one feature (e.g. gender, as in the previous example), clearly share common coding principles with the parallel representations that we propose. Moreover, this type of vector representation has been shown to be highly predictive of fMRI BOLD signals measured while subjects are presented with semantically meaningful stimuli [30].

A different but also very appealing approach to extracting compositional features in an unsupervised way relies on variational Bayesian inference to learn to infer interpretable factorized representations of some inputs [31–34]. These methods have exhibited remarkable success in disentangling independent factors of variations of a variety of real-world datasets, and it will be exciting in the future to examine whether our methods will have any bearing on the analysis of the functioning of these algorithms.

Abstraction is also an important active area of research in Reinforcement Learning (RL), as it is a fertile concept for solution strategies to cope with the notorious "curse of dimensionality", i.e. the exponential growth of the solution space of a problem with the size of the encoding of its states [35]. Most abstraction techniques in RL can be divided in two main categories: 'temporal abstraction' and 'state abstraction'.

Temporal abstraction is the workhorse of Hierarchical Reinforcement Learning [36–38] and is based on the notion of temporally extended actions (or options): the idea of enriching the repertoire of actions available to the agent with "macro-actions" composed of conditional sequences of atomic actions built to achieve useful sub-goals in the environment. Temporal abstraction can be thought of as an attempt to reduce the dimensionality of the space of action sequences: instead of having to compose policies in terms of long sequences of actions, the agent can select options that automatically extend for several time steps.

State abstraction methods are most closely related to our work. In brief, they rely on the idea of simplifying the representation of the domain exposed to the agent by hiding or removing information about the environment that is non-critical to maximize the reward function. Typical instantiations of this technique involve information hiding, clustering of states, and other forms of domain aggregation and reduction [39]. The use of neural networks as function approximators to represent a decision policy that we used in the last section of the Results and in Supplementary Information S3 also falls within the category of a state abstraction method. The idea is that the inductive bias of neural networks induces generalization across similar inputs, allowing them to mitigate the curse of dimensionality in high-dimensional domains. A particularly well-known success story of neural networks applied to reinforcement learning is Deep Q-learning[16], which employed a deep neural network representation of the Q-function of an agent trained to play 49 different Atari video games with super-human performance using a combination of temporal-difference learning and back-propagation. Our efforts to model the task used in the experiments in section S3 demonstrates that neural networks induce the type of abstract representations that we propose

and suggests that our analysis techniques could be useful to elucidate the geometric properties underlying the success of Deep Q-learning neural networks.

Characterizing brain areas by analyzing the geometry of neural representations Historically, brain areas have been characterized by describing what task-relevant variables are encoded, and by relating the encoding of these variables to behavior, either by correlating neural activity with behavioral measures, or by perturbing neural activity to assess the necessity or sufficiency of the signals provided by the brain area. Here we describe a geometric characterization of neural activity patterns that goes beyond variable encoding and corresponds to a notion of abstraction defined in terms of generalization across different instances of the same variable. As discussed previously, even random patterns can encode task-relevant variables, but without representing them in an abstract format. Consequently, merely requiring that a variable be encoded by a neural population may fail to identify those structures responsible for cognitive functions that demand utilization of information stored in an abstract format. The analysis of geometric features such as the parallelism score of the activity patterns evoked in different neural populations could thereby reveal important functional differences between brain areas, differences not apparent from a decoding analysis alone or from an analysis of single neuron response properties.

The generation of neural representations of variables in an abstract format is central to many different sensory, cognitive and emotional functions. For example, in vision, the creation of neural representations of objects that are invariant with respect to their position, size and orientation in the visual field is a typical abstraction process that has been studied in machine learning applications (see e.g. [40,41]) and in the brain areas involved in representing visual stimuli (see e.g. [42,43]). This form of abstraction may underlie fundamental aspects of perceptual learning. Here we have focused on a form of abstraction that we believe is essential to higher cognitive functions, such as context-dependent decision-making and emotion regulation, the use of conceptual reasoning to learn from experience, and the application of inference. The types of abstraction that underlie these processes almost certainly rely on reinforcement learning and memory, as well as the ability to forge conceptual links across category boundaries. The analysis tools developed here can be applied to electrophysiological, fMRI and calcium imagining data and may prove valuable for understanding how different brain areas contribute to various forms of abstraction that underlie a broad range of mental functions. Future studies must focus on the specific neural mechanisms that lead to the formation of abstract representations, which is fundamentally important for any form of learning, for executive functioning, and for cognitive and emotional flexibility.

Acknowledgements We are grateful to L.F. Abbott and R. Axel for many useful comments on the manuscript. This project is supported by the Simons Foundation, and by NIMH (1K08MH115365, R01MH082017). SF and MKB are also supported by the Gatsby Charitable Foundation, the Swartz Foundation, the Kavli foundation and the NSF's NeuroNex program award DBI-1707398. JM is supported by the Fyssen Foundation. SB received support from NIMH (1K08MH115365, T32MH015144 and R25MH086466), and from the American Psychiatric Association and Brain & Behavior Research Foundation young investigator fellowships.

Competing Interests The authors declare that they have no competing financial interests.

Correspondence Correspondence and requests for materials should be addressed to S. Fusi or to D. Salzman (email: sf2237@columbia.edu, cds2005@columbia.edu).

1. Altmann, G. Abstraction and generalization in statistical learning: implications for the relationship between semantic types and episodic tokens. *Philos Trans R Soc Lond B Biol Sci.* **372**, 20160060 (2017).

2. Milner, B., Squire, L. & Kandell, E. Cognitive neuroscience and the study of memory.. *Neuron* **1998**, 445–468 (1998).

3. Eichenbaum, H. Hippocampus: Cognitive processes and neural representations that underlie declarative memory. *Neuron* **2004**, 109–120 (2004).

4. Wirth, S. *et al.* Single neurons in the monkey hippocampus and learning of new associations. *Science* **300**, 1578–1581 (2003).

5. Schapiro, A. C., Turk-Browne, N. B., Norman, K. A. & Botvinick, M. M. Statistical learning of temporal community structure in the hippocampus. *Hippocampus* **26**, 3–8 (2016).

6. Wallis, J. D., Anderson, K. C. & Miller, E. K. Single neurons in prefrontal cortex encode abstract rules. *Nature* **411**, 953 (2001).

7. Miller, E. K., Nieder, A., Freedman, D. J. & Wallis, J. D. Neural correlates of categories and concepts. *Current opinion in neurobiology* **13**, 198–203 (2003).

8. Buckley, M. J. *et al.* Dissociable components of rule-guided behavior depend on distinct medial and prefrontal regions.. *Science* **325**, 52–58 (2009).

9. Antzoulatos, E. G. & Miller, E. K. Differences between neural activity in prefrontal cortex and striatum during learning of novel abstract categories. *Neuron* **71**, 243–249 (2011).

10. Wutz, A., Loonis, R., Roy, J. E., Donoghue, J. A. & Miller, E. K. Different levels of category abstraction by different dynamics in different prefrontal areas. *Neuron* **97**, 716–726 (2018).

11. Saez, A., Rigotti, M., Ostojic, S., Fusi, S. & Salzman, C. Abstract context representations in primate amygdala and prefrontal cortex. *Neuron* **87**, 869–881 (2015).

12. Chung, S., Lee, D. D. & Sompolinsky, H. Classification and geometry of general perceptual manifolds. *Physical Review X* **8**, 031003 (2018).

13. Rigotti, M. *et al.* The importance of mixed selectivity in complex cognitive tasks. *Nature* **497**, 585 (2013).

14. Fusi, S., Miller, E. K. & Rigotti, M. Why neurons mix: high dimensionality for higher cognition. *Current opinion in neurobiology* **37**, 66–74 (2016).

15. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).

16. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529 (2015).

17. Yakovlev, V., Fusi, S., Berman, E. & Zohary, E. Inter-trial neuronal activity in inferior temporal cortex: a putative vehicle to generate long-term visual associations. *Nature neuroscience* **1**, 310 (1998).

18. Bernacchia, A., Seo, H., Lee, D. & Wang, X.-J. A reservoir of time constants for memory traces in cortical neurons. *Nature neuroscience* **14**, 366 (2011).

19. Akrami, A., Kopec, C. D., Diamond, M. E. & Brody, C. D. Posterior parietal cortex represents sensory history and mediates its effects on behaviour. *Nature* **554**, 368 (2018).

20. Miller, E. K. & Cohen, J. D. An integrative theory of prefrontal cortex function. *Annual review of neuroscience* **24**, 167–202 (2001).

21. Kane, M. J. & Engle, R. W. The role of prefrontal cortex in working-memory capacity, executive attention, and general fluid intelligence: An individual-differences perspective. *Psychonomic bulletin & review* **9**, 637–671 (2002).

22. Curtis, C. E. & D'Esposito, M. Persistent activity in the prefrontal cortex during working memory. *Trends in cognitive sciences* **7**, 415–423 (2003).

23. Stringer, C., Pachitariu, M., Steinmetz, N., Carandini, M. & Harris, K. D. High-dimensional geometry of population responses in visual cortex. *bioRxiv* 374090 (2018).

24. Gao, P. & Ganguli, S. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current opinion in neurobiology* **32**, 148–155 (2015).

25. Mazzucato, L., Fontanini, A. & La Camera, G. Stimuli reduce the dimensionality of cortical activity. *Frontiers in systems neuroscience* **10**, 11 (2016).

26. Tang, E., Mattar, M. G., Giusti, C., Thompson-Schill, S. L. & Bassett, D. S. Effective learning is accompanied by increasingly efficient dimensionality of whole-brain responses. *arXiv preprint arXiv:1709.10045* (2017).

27. Lindsay, G. W., Rigotti, M., Warden, M. R., Miller, E. K. & Fusi, S. Hebbian learning in a random network captures selectivity properties of the prefrontal cortex. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **37**, 11021–11036 (2017).

28. Mikolov, T., Yih, W.-t. & Zweig, G. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 746–751 (2013).

29. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119 (2013).

30. Mitchell, T. M. *et al.* Predicting human brain activity associated with the meanings of nouns. *science* **320**, 1191–1195 (2008).

31. Chen, X. *et al.* Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, 2172–2180 (2016).

32. Higgins, I. *et al.* β-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR* (2017).

33. Chen, T. Q., Li, X., Grosse, R. B. & Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems 31*, 2614–2624 (2018).

34. Kim, H. & Mnih, A. Disentangling by factorising. *arXiv preprint arXiv:1802.05983* (2018).

35. Bellman, R. E. *Dynamic Programming.* (Princeton University Press, 1957).

36. Dietterich, T. G. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research* **13**, 227–303 (2000).

37. Precup, D. *Temporal abstraction in reinforcement learning* (PhD thesis, University of Massachusetts Amherst, 2000).

38. Barto, A. G. & Mahadevan, S. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems* **13**, 341–379 (2003).

39. Ponsen, M., Taylor, M. E. & Tuyls, K. Abstraction and generalization in reinforcement learning: A summary and framework. In *International Workshop on Adaptive and Learning Agents*, 1–32 (Springer, 2009).

40. Riesenhuber, M. & Poggio, T. Hierarchical models of object recognition in cortex. *Nature neuroscience* **2**, 1019 (1999).

41. LeCun, Y., Bengio, J. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).

42. Freedman, D. J., Riesenhuber, M., Poggio, T. & Miller, E. K. Categorical representation of visual stimuli in the primate prefrontal cortex. *Science* **291**, 312–316 (2001).

43. Rust, N. & Dicarlo, J. Selectivity and tolerance ("invariance") both increase as visual information propagates from cortical area v4 to it. *J Neurosci* **30**, 12978–12995 (2010).

44. Meyers, E. M., Freedman, D. J., Kreiman, G., Miller, E. K. & Poggio, T. Dynamic population coding of category information in inferior temporal and prefrontal cortex. *Journal of neurophysiology* **100**, 1407 (2008).

45. Golland, P., Liang, F., Mukherjee, S. & Panchenko, D. Permutation tests for classification. In *International Conference on Computational Learning Theory*, 501–515 (Springer, 2005).

46. Stefanini, F. *et al.* A distributed neural code in ensembles of dentate gyrus granule cells. *bioRxiv* 292953 (2018).

47. Morcos, A. S., Barrett, D. G., Rabinowitz, N. C. & Botvinick, M. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959* (2018).

48. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

49. Paszke, A. *et al.* Automatic differentiation in pytorch (2017).

50. Rigotti, M., Ben Dayan Rubin, D., Wang, X.-J. & Fusi, S. Internal representation of task rules by recurrent dynamics: the importance of the diversity of neural responses. *Frontiers in Computational Neuroscience* **4**, 24 (2010).

51. Barak, O., Rigotti, M. & Fusi, S. The sparseness of mixed selectivity neurons controls the generalization-discrimination trade-off. *The Journal of neuroscience : the official journal of the Society for Neuroscience* **33**, 3844–3856 (2013).

52. Machens, C. K., Romo, R. & Brody, C. D. Functional, but not anatomical, separation of "what" and "when" in prefrontal cortex. *J Neurosci* **30**, 350–360 (2010). URL `http://dx.doi.org/10.1523/JNEUROSCI.3276-09.2010`.

53. Chen, X. Confidence interval for the mean of a bounded random variable and its applications in point estimation. *arXiv preprint arXiv:0802.3458* (2008).

Methods

M1 Task and Behavior

Two rhesus monkeys (Macaca mulatta; two males respectively, 8 and 13 kg) were used in these experiments. All experimental procedures were in accordance with the National Institutes of Health guide for the care and use of laboratory animals and the Animal Care and Use Committees at New York State Psychiatric Institute and Columbia University. Monkeys performed a serial-reversal learning task in which they were presented one of four visual stimuli (fractal patterns). Stimuli were consistent across contexts and sessions, presented in random order. Each trial began with the animal holding down a button and fixating for 400 ms (Fig. 1a). If those conditions were satisfied, one of the four stimuli was displayed on a screen for 500 ms. In each context, correct performance for two of the stimuli required releasing the button within 900 ms of stimulus disappearance; for the other two, the correct operant action was to continue to hold the button. For 2 of the 4 stimuli, correct performance resulted in reward delivery; for the other 2, correct performance did not result in reward. If the monkey performed the correct action, a trace interval of 500 ms ensued followed by the liquid reward or by a new trial in the case of non-rewarded stimuli. If the monkey made a mistake, a 500 ms time out was followed by the repetition of the same trial type if the stimulus was a non-rewarded one. In the case of incorrect responses to rewarded stimuli, the time-out was not followed by trial repetition and the monkey simply lost his reward. After a random number of trials between 50 and 70, the context switched without warning and with it the operant and reinforcement contingencies changed. Operant contingencies switched for all images, but for two stimuli the reinforcement contingencies did not change, in order to ensure orthogonality between operant and reinforcement contingencies. A different colored frame (red or blue) for each context appears on the edges of the monitor on 10 percent of the trials, randomly selected, and only on specific stimulus types (stimulus C for context 1 and stimulus D for context 2) although never in the first five trials following a contextual switch. All trials with a contextual frame were excluded from all analyses presented.

M2 Electrophysiological Recordings

Recordings began only after the monkeys were fully proficient in the task and performance was stable. Recordings were conducted with multi-contact vertical arrays electrodes (v-probes, Plexon Inc., Dallas, TX) with 16 contacts spaced at 100 μm intervals in ACC and DLPFC, and 24 contacts in HPC, using the Omniplex system (Plexon Inc.). In each session, we individually advanced the arrays into the three brain areas using a motorized multi-electrode drive (NAN Instruments). Analog signals were amplified, band-pass filtered (250 Hz - 8 kHz), and digitized (40 kHz) using a Plexon MAP system (Plexon, Inc.). Single units were isolated offline using Plexon Offline Sorter. To address the possibility that overlapping neural activity was recorded on adjacent contacts, or that two different clusters visible on PCA belonged to the same neuron, we compared the zero-shift cross-correlation in the spike trains with a 0.2 ms bin width of each neuron identified in the

same area in the same session. If 10 percent of spikes co-occurred, the clusters were considered duplicated and one was eliminated. If 1-10 percent of spikes co-occurred, the cluster was flagged and isolation was checked for a possible third contaminant cell. Recording sites in DLPFC were located in Brodmann areas 8, 9 and 46. Recording sites in ACC were in the ventral bank of the ACC sulcus (area 24c). HPC recordings were largely in the anterior third, spanning across CA1-CA2-CA3 and DG.

M3 Selection of trials/neurons, and decoding analysis:

The neural population decoding algorithm was based on a linear classifier (see e.g. [11]) trained on pseudo-simultaneous population response vectors composed of the spike counts of the recorded neurons within specified time bins and in specific trials [44]. The trials used in the decoding analysis are only those in which the animal responded correctly (both for the current trial and the directly preceding one), in which no context frame was shown (neither during the current nor the preceding trial), and which occurred at least five trials after the most recent context switch. We retain all neurons for which we have recorded at least 15 trials satisfying these requirements for each of the eight experimental conditions (i.e. the combinations of context, value and action). Every decoding analysis is averaged across several repetitions to estimate trial-to-trial variability (as explained more in detail below). For every repetition, from among all selected trials we randomly split off five trials per condition to serve as our test set, and used the remaining trials (at least ten per condition) in our training set. For every neuron and every time bin, we normalized the distribution of spike counts across all trials in all conditions with means and standard deviations computed on the trials in the training set. Specifically, given an experimental condition c (i.e. a combination context, value and action) in a time bin t under consideration, we generated the pseudo-simultaneous population response vectors by sampling, for every neuron i, the z-scored spike count in a randomly selected trial in condition c, which we indicate by $n_i^c(t)$. This resulted in a single-trial population response vector $n^c(t) = (n_1^c(t), n_2^c(t), \ldots, n_N^c(t))$, where N corresponds to the number of recorded neurons in an area under consideration. This single-trial response vector can be thought of as a noisy measurement of an underlying mean firing rate vector $\bar{n}^c(t)$, such that $n^c(t) = \bar{n}^c(t) + \eta^c(t)$, with $\eta^c(t)$ indicating a noise vector modeling the trial-to-trial variability of spike counts. Assuming that the trial-to-trial noise is centered at zero, we estimate the mean firing rate vectors taking the sample average: $\bar{n}^c(t) \approx \langle n^c(t) \rangle$, where the angular brackets indicate averaging across trials. We then either trained maximum margin (SVM) linear classifiers on the estimated mean firing rate vectors for the eight conditions in the training set, or we trained such classifiers on the single-trial population response vectors generated from the training set of trials. In the latter case, in order to obtain a number of trials that is large compared to the number of neurons, we re-sampled the noise by randomly picking noisy firing rates (i.e. spike counts) from among all the training trials of a given experimental condition for each neuron independently. Specifically, we re-sampled 10,000 trials per condition from the training set. While this neglected correlations between different neurons within conditions, we only had little information about these correlations in the first place, since only a relatively small numbers of neurons were recorded simultaneously. Regardless of whether

we trained on estimated mean firing rate vectors or on re-sampled single-trial population response vectors, the decoding performance is measured on a cross-validated manner on 1000 resampled single-trial population response vectors generated from the test set of trials. For every decoding analysis training and testing were then repeated 1,000 times over different random partitions of the trials into training and test trials. The decoding accuracy that we reported were computed as the average results across repetitions.

Statistical significance of the decoding accuracy was assessed using a permutation test for classification[45]. Specifically, we repeated the same procedure just described, but at the beginning of every repetition of a decoding analysis, trials were shuffled, i.e. associated to a random condition. This is a way of estimating the probability that the population decoders that we used would have given the same results that we obtained by chance, i.e. when applied on data that contain no information regarding the experimental conditions.

In Fig. S2 we show the cross-validated decoding accuracy as a function of time throughout the trial (for a sliding 500 ms time window) for maximum margin classifiers trained only on the mean neural activities for each condition, while Fig. 4c shows similar results for linear classifiers trained on the mean firing rates in the neural data within a time window from -800 ms to 100 ms relative to stimulus onset.

For all analyses, data were combined across monkeys, because all key features of the data set were consistent across the two monkeys.

M4 The cross-condition generalization performance (CCGP):

The hallmark feature of abstract neural representations is their ability to support generalization. When several abstract (in our case binary) variables are encoded simultaneously, generalization must be possible for all the abstract variables. We quantify a powerful form of generalization using a measure we call the cross-condition generalization performance (in fact we can view this measure as a quantitative definition of the degree of abstraction of a set of neural representations). It is analogous to the cross-validated decoding performance commonly employed, except that instead of splitting up the data randomly, such that trials from all conditions will be present in both the training and test sets, we instead perform the split according to the condition labels, such that the training set consists entirely of trials from one group of conditions, while the test set consists only of trials from a disjoint group of conditions. We train (on the former) a linear classifier for a certain dichotomy that discriminates the conditions in the training set according to some label (one of the abstract variables), and then ask whether this discrimination generalizes to the test set by measuring the classification performance on the data from entirely different conditions, which were never seen during training.

This means that in order to achieve a large cross-conditions generalization performance, it is not sufficient to merely generalize over the noise associated with trial-to-trial fluctuations of the neural activity around the mean firing rates corresponding to individual conditions. Instead, the classifier has to generalize also across different conditions on the same side of an (abstract) dichotomy, i.e., across those conditions that belong to the same category according to the abstract

variable under consideration.

Given our experimental design with eight different conditions (distinguished by context, value and action of the previous trial), we can investigate different balanced (four versus four condition) dichotomies, and choose one, two or three conditions from each side of a dichotomy to form our training set. We use the remaining conditions (three, two or one from either side, respectively) for testing, with larger training sets typically leading to better generalization performance. For different choices of training conditions we will in general obtain different values of the classification performance on the test conditions, and we define the cross-condition generalization performance (CCGP) as its average over all possible sets of training conditions (of a given size). In Fig. 4a we show the CCGP (on the held out fourth condition) when training on three conditions from either side of the context, value or action dichotomies.

The selection of trials used is the same as for the decoding analysis, except that here we retain all neurons that have at least ten trials for each experimental condition that meet our selection criteria (since the split into training and test sets is determined by the labels of the eight conditions themselves, so that for a training condition we don't need to hold out additional test trials). We pre-process the data by z-scoring each neuron's spike count distribution separately. Again, we can either train a maximum margin linear classifier only on the cluster centers, or on the full training set with trial-to-trial fluctuations (noise), in which case we re-sample 10,000 trials per condition, with Fig. 4a showing results using the latter method.

M5 The parallelism score (PS):

We developed a measure based on angles of coding directions to characterize the geometry of neural representations of variables in the firing rate space. When training a linear classifier on a pair of conditions (one from each side of a dichotomy) that differ only in one label (but agree on all others), the weight vector defining the resulting separating hyperplane will be aligned with the vector connecting the cluster centers corresponding to the neural representations of the two training conditions if we assume isotropic noise around both of them. This corresponds to the coding direction for the potentially abstract variable under consideration, given this choice of training set. Other coding directions for the same variable can be obtained by choosing a different pair of training conditions (defined by different, but equal values for the other variables corresponding to orthogonal dichotomies). The separating hyperplane associated with one such pair of training conditions is more likely to correctly generalize to another pair of conditions if the associated coding directions are parallel (as illustrated in Fig. 2d,e). Therefore, we introduce a measure to quantify the alignment of the different coding directions, which we call the parallelism score (PS).

If we had only four conditions (and hence at most two abstract variables) as shown in Fig. 2d, there would be only two coding directions for a given variable (from the two pairs of training conditions), and we would simply consider the cosine of the angle between them (i.e., the normalized overlap of the two weight vectors). In our experiments, there were eight conditions (leading to at most three perfectly abstract variables), and thus pairing them across the separating hyperplane will lead to four normalized coding vectors $\vec{v_i}$ for $i = 1, 2, 3, 4$ (corresponding to four pairs of

training conditions). In this case, we consider the cosines of the angles between two of them $\cos(\theta_{ij}) = \vec{v_i}.\vec{v_j}$, and we average over all six of these angles (corresponding to all possible choices of two different coding vectors). Note that these coding directions are simply the unit vectors pointing from one cluster center to another, and we don't train any classifiers for this analysis.

In general there are multiple ways of pairing up conditions across the separating hyperplane of a dichotomy under consideration. Because we don't want to assume a priori that we know the correct way of pairing up conditions (which would depend on the labels of the other abstract variables), we instead consider all possible ways of matching up the conditions on the two sides of the dichotomy one-to-one, and then define the PS as the maximum across all possible pairings of the average cosine. There are two such pairings in the case of four conditions, and 24 pairings for eight conditions (in general there are $(m/2)!$ for m conditions, so there would be a combinatorial explosion if m was large). Therefore, the parallelism score (for eight conditions) is defined as

$$\text{Parallelism Score} = \max_{\text{pairings of conditions}} \sum_{i=1}^{4} \sum_{j>i}^{4} \cos(\theta_{ij})/6 \ . \tag{M1}$$

The parallelism scores of the context, value and action dichotomies of our data are plotted in Fig. 4b. The selection of trials used in this analysis is the same as for the decoding and cross-condition generalization analyses, retaining all neurons that have at least ten trials for each experimental condition that meet our selection criteria, and z-scoring each neuron's spike count distribution individually.

Note that a high parallelism score for one variable/dichotomy doesn't necessarily imply perfect generalization across other variables. Even if the coding vectors for a given variable are approximately parallel, the test conditions might be much closer together than the training conditions. In this case generalization would likely be poor and the orthogonal dichotomy would have a low parallelism score. (Moving the cluster centers in neural representation space affects the parallelism scores of at least some dichotomies, and despite being based on angles the set of all such scores depends implicitly on pairwise distances, except on the overall scale of the whole geometry). Even high parallelism scores for multiple variables don't guarantee good generalization of one dichotomy across another one. When training a linear classifier on noisy data, the shape of the noise clouds could skew the weight vector of a maximum margin classifier away from the vector connecting the cluster centers of the training conditions. In addition, even if this is not the case and the noise is isotropic, generalization might still fail because of a lack of orthogonality of the coding directions for different variables (the eight conditions might be arranged at the corners of a parallelepiped instead of a cuboid). In summary, while the parallelism score is not equivalent to the cross-condition generalization performance, high scores for a number of dichotomies with orthogonal labels characterize a family of (approximately factorizable) geometries that can lead to good generalization properties if the noise is sufficiently well behaved (consider e.g. the case of the principal axes of the noise distributions being aligned with the coding vectors), and specifically for the simple case of isotropic noise, if the coding directions for different variables are approximately orthogonal to each other.

M6 Random models:

In order to assess the statistical significance of the above analyses we need to compare our results (for the decoding performance, abstraction index, cross-condition generalization performance, and parallelism score, which we collectively refer to as scores here) to the distribution of values expected from an appropriately defined random control model. There are various sensible choices for such random models, each corresponding to a somewhat different null hypothesis we might want to reject. The simplest case we consider is a shuffle of the data, in which assign a new, random condition label to each trial for each neuron independently (in a manner that preserves the total number of trials for each condition). When re-sampling artificial, noisy trials, we shuffle first, and then re-sample in a manner that respects the new, random condition labels. This procedure destroys almost all structure in the data, except the marginal distributions of firing rates of individual neurons. The error bars around chance level for the decoding performance in Figs. S2 and 4, and for the parallelism score in Figs. 4 and 5, are based on this shuffle control (plus/minus two standard deviations).

A different kind of structure is retained in a class of geometric random models, which we construct in order to rule out another type of null hypothesis. For the analyses that depend only on the cluster centers of the eight conditions, we can construct a random geometry by sampling new cluster centers from an isotropic Gaussian distribution (and rescaling it to keep the total signal variance the same as in the data). Such a random arrangement of the mean firing rates (cluster centers) is a very useful control to compare against, since such geometries do not constitute abstract neural representations, but nevertheless typically allow relevant variables to be decoded. For analyses that depend also on the structure of the noise (in particular, decoding and CCGP with re-sampled trials), our random model in addition requires some assumptions about the noise distributions. We could simply choose identical isotropic noise distributions around each cluster center, but training a linear classifier on trials sampled from such a model would essentially be equivalent to training a maximum margin classifier on the cluster centers only. Instead, we choose to preserve some of the noise structure of the data by moving the (re-sampled) noise clouds to the new random position of the corresponding cluster and performing a discrete rotation around it by permuting the axes (for each condition independently). If our scores are significantly different from those obtained using this random model, we can reject the null hypothesis that the data was generated by a random isotropic geometry with the same total signal variance and similarly shaped noise clouds as in the data. The error bars around chance level for the CCGP in Figs. 4 and 5 are derived from this geometric random control model.

We can also consider the distribution of scores across the 35 different balanced dichotomies we can form using the eight conditions in our data set. Since there are clearly correlations between the scores of different dichotomies (e.g. because the labels may be partially overlapping, i.e., not orthogonal), we do not think of this distribution as a random model to assess the probability of obtaining certain scores from unstructured data. However, it does allow us to make statements about the relative magnitude of the scores compared to those of other variables that may also be decodable from the data and possibly abstract.

M7 Simulations of the multi-layer network:

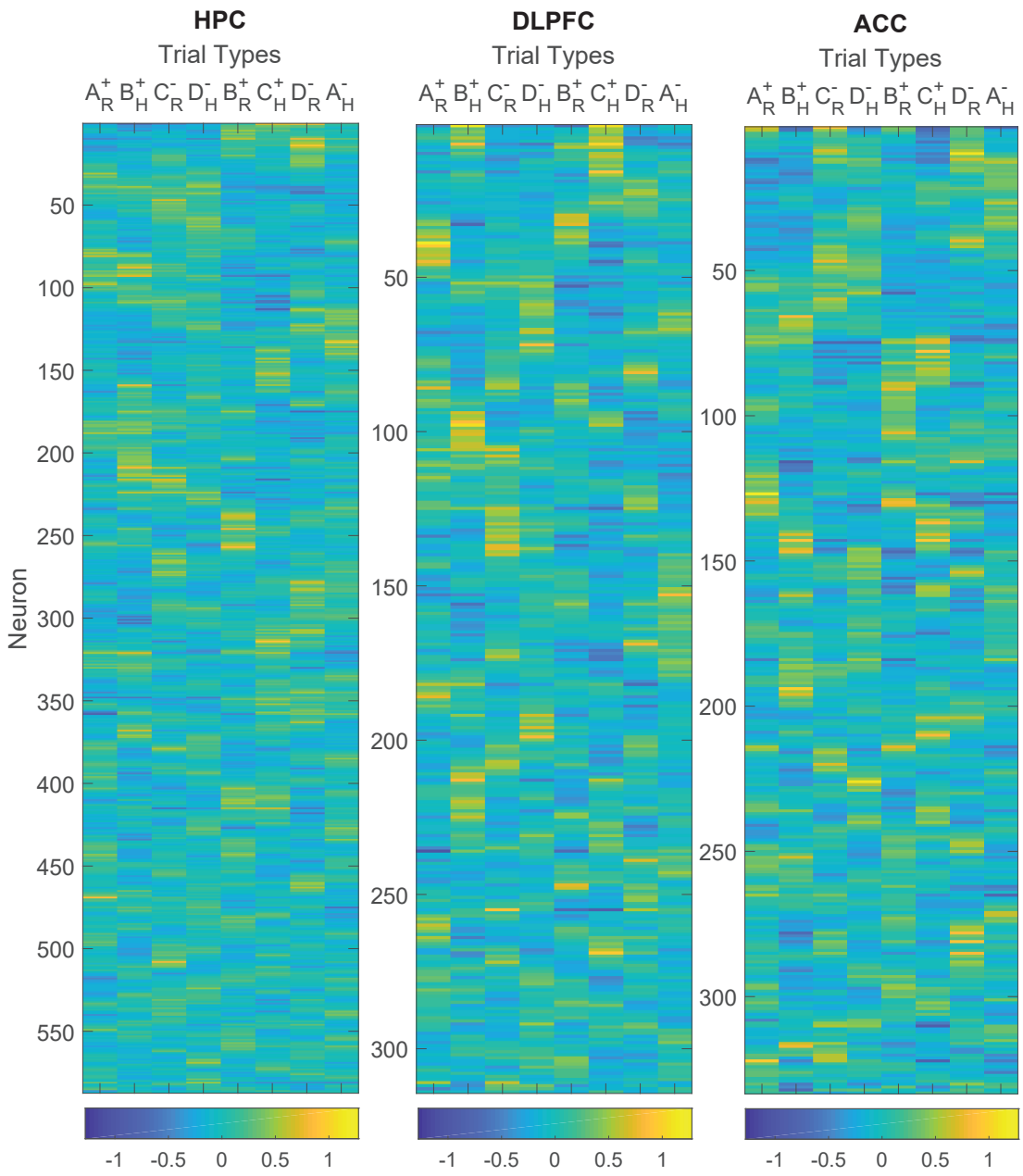
The two hidden layer network depicted in Figure 5 contains 768 neurons in the input layer, 100 in each hidden layer and four neurons in the output layer. We used eight digits (1-8) of the full MNIST data set to match the number of conditions we considered in the analysis of the experiment. The training set contained 48128 images and the test set contained 8011 digits. The network was trained to output the parity and the magnitude of each digit and to report it using four output units: one for odd, one for even, one for small (i.e. a digit smaller than 5) and one for large (a digit larger than 4). We trained the network using the back-propagation algorithm '`train`' of matlab (with the neural networks package). We used a tan-sigmoidal transfer function ('`tansig`' in matlab), the mean squared normalized error ('`mse`') as the cost function, and the maximum number of training epochs was set to 400. After training, we performed the analysis of the neural representations using the same analytical tools that we used for the experimental data, except that we did not z-score the neural activities since they were simultaneously observed in the simulations.

Supplementary Information

S1 The abstraction index

A simple way to achieve an abstract neural representation would be to cluster together the activity patterns corresponding to the conditions on either side of a certain dichotomy (defining a binary variable). For example, if the spike count patterns in a certain brain area were equal for all four stimuli in context one, and similarly coincided (at a different value) for context two, this area would exhibit an abstract representation of context, at the expense of not encoding any information about stimulus identity, operant action or reward value. We can assess the degree of clustering by comparing the average distance between the mean neural activity patterns corresponding to the conditions on the same side of a dichotomy (within or intra-group) to the average distance between those on opposite sides of the dichotomy (between or inter-group). For balanced (four versus four)

Figure S1 *(following page)*: The activity of all the neurons used for the abstraction analyses (i.e., all the neurons recorded for at least ten trials per condition after excluding: 1) those trials performed incorrectly; 2) those trials in which a contextual frame was shown either for the current or the previous trial; and 3) those trials that occurred less than five trials after a context switch). The z-scored firing rates are estimated in the 900 ms time window that starts 800ms before stimulus onset. Each row represents the activity of an individual neuron. Different columns correspond to different conditions (i.e., the stimulus-action-outcome sequence preceding the interval). Neurons are ordered such that for adjacent rows, the eight-dimensional vectors of z-scored activities point in similar directions. The responses are very diverse in all three brain areas.


HPC
Trial Types
A_R^+ B_H^+ C_R^- D_H^- B_R^+ C_H^+ D_R^- A_H^-
Neuron
50
100
150
200
250
300
350
400
450
500
550
-1 -0.5 0 0.5 1
DLPFC
Trial Types
A_R^+ B_H^+ C_R^- D_H^- B_R^+ C_H^+ D_R^- A_H^-
50
100
150
200
250
300
-1 -0.5 0 0.5 1
ACC
Trial Types
A_R^+ B_H^+ C_R^- D_H^- B_R^+ C_H^+ D_R^- A_H^-
50
100
150
200
250
300
-1 -0.5 0 0.5 1

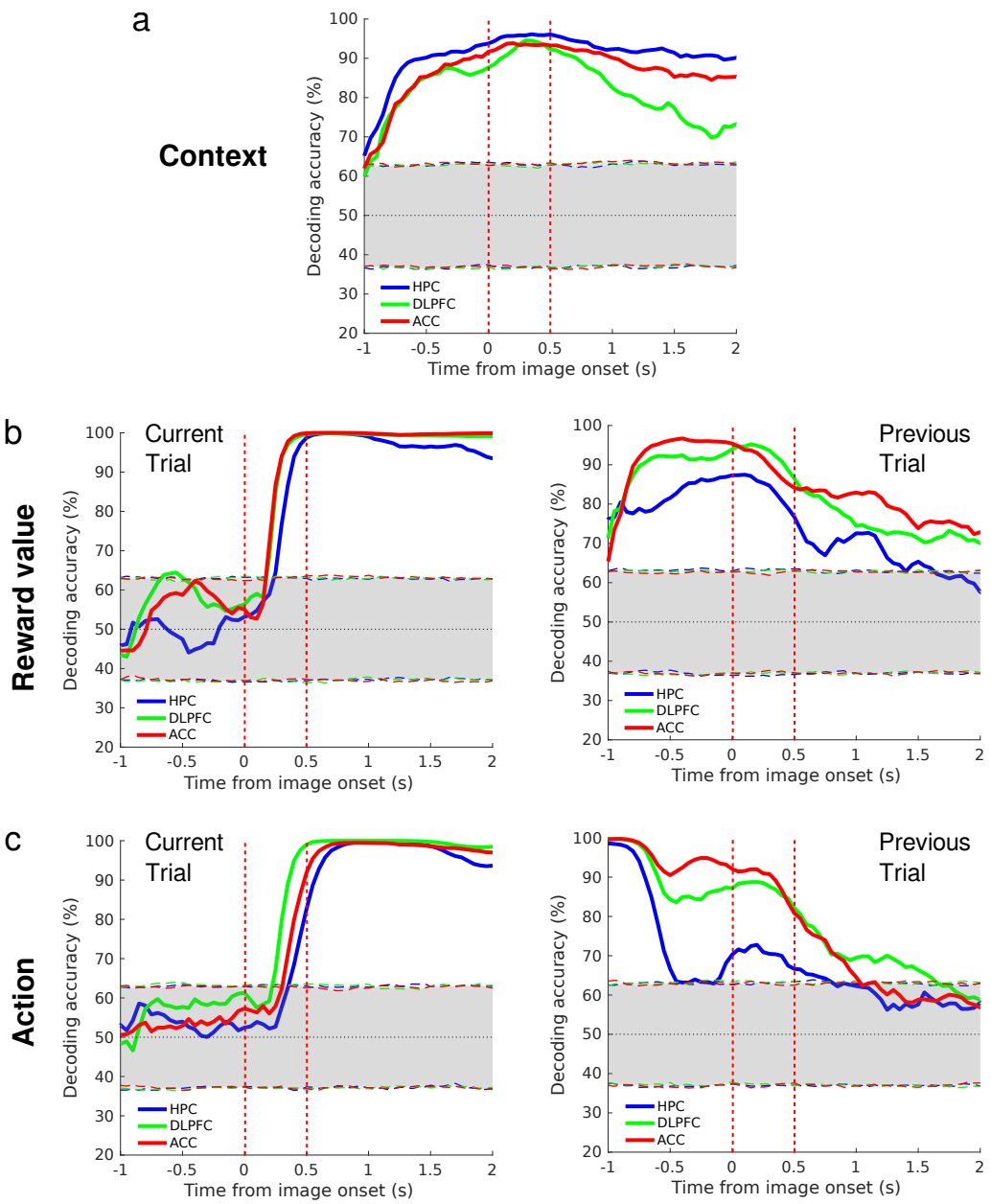


Figure S2: Population level encoding of task-related variables. a-e. Performance of a linear decoder plotted a function of time relative to image onset for classifying a task-relevant variable. a. Context on the current trial. b. Reinforcement outcome on current (left) and prior (right) trials. c. Operant action on current (left) and prior (right) trials. The decoding performance was computed in a 500-ms sliding window stepped every 50 ms across the trial for the three brain areas separately (blue, HPC; red, ACC; green, DLPFC). Dashed lines indicate two standard deviations above and below the mean of the distribution of accuracies obtained with a permutation test done by shuffling trials 1000 times (bootstrap). The image is displayed on the screen from time 0 to 0.5 sec. Analyses were run only on correct trials at least 5 trials after a context switch.

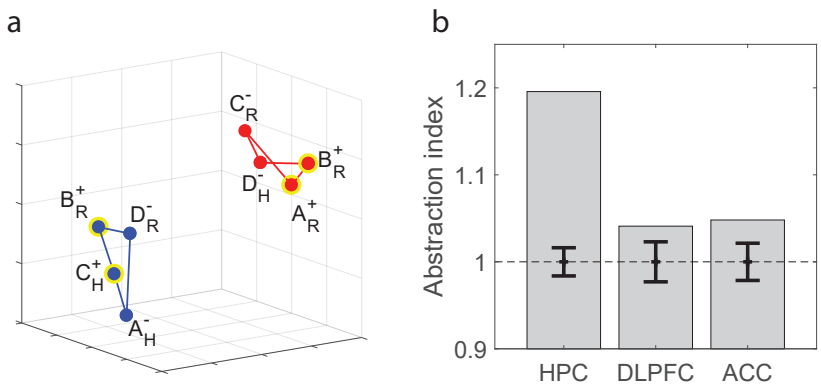


Figure S3: Abstraction by clustering. a) Schematic of firing rate space in the case of clustering abstraction (as in Figure 2a). Due to the clustering, the average within-context distance is shorter than the mean between-context distance. b) Abstraction index for the context dichotomy (ratio of average between-context distance to average within-context distance using a simple Euclidean metric) for the z-scored neural firing rates recorded from HPC, DLPFC and ACC, averaged over a time window of -800ms to 100ms relative to stimulus onset. The error bars are plus/minus two standard deviations around chance level (unit abstraction index), obtained from a shuffle of the data.

dichotomies of the eight experimental conditions (context, value and action of the previous trial), there are 16 inter-group distances, and 12 intra-group distances (six on each side of the dichotomy) that contribute. We define the ratio of the average between group distance to the average within group distance as the abstraction index, which measures the degree of clustering of a set of neural representations associated with a certain dichotomy. In the absence of any particular geometric structure (such as clustering), we would expect these two average distances to be equal, resulting in an abstraction index of one, while clustering would lead to values larger than one.

Fig. S3 shows the abstraction index for the context variable computed from the measured neural activity patterns. As above for decoding, we retain only correct trials without a contextual frame that didn't occur within 5 trials of a context switch for this analysis. We z-score the overall activity distribution of each neuron before computing the mean activity pattern of each condition, and use a simple Euclidean distance metric (employing a Mahalanobis distance metric instead yields similar results).

S2 The recorded neurons are not highly specialized

An ensemble of neurons could in principle encode multiple abstract variables simply by assigning each neuron to be tuned to precisely one of these variables. In this case, the ensemble can be divided into a number of subpopulations each of which exhibits pure selectivity for one of the abstract variables. Geometrically, this situation is similar to the one depicted in Fig. 2a. The situation in which neurons exhibit mixed selectivity can be obtained by rotating this geometry, as shown in

Fig. 2b. This rotated representation would show the same generalization properties. In the data, we do not observe many pure selectivity neurons. If neurons were highly specialized for particular variables, training a linear classifier to decode that variable should lead to very large readout weights from the associated specialized subpopulation, but very small readout weights from other neurons (which might specialize on encoding other variables). Therefore, in a scatter plot of the (absolute values) of the readout weights for different classifiers we would expect specialized neurons to fall close to the axes (with large weights for the preferred variable, but small ones for any others). If this situation occurred for a large number of neurons, we might expect a negative correlation between the absolute values of the decoding weights for different variables. However, in fact the weights do not cluster close to the axes, as shown in Fig. S4. The correlation coefficients of their absolute values are positive. We conclude that highly specialized neurons are not particularly common in the neural ensembles we recorded, i.e. pure selectivity appears to be no more likely than coding properties corresponding to a random linear combination of the task-relevant variables.

For each neuron we can consider the three-dimensional space of the readout weights for the three variables (components of three different unit-norm weight vectors in the space of neural activities). Clearly neurons with large readout weights for the linear classifier trained to read out a particular variable are important for decoding that variable. Some neurons have larger total readout weights than others (e.g. we can consider the sum of squares of the three weight components, corresponding to the squared radius in the three-dimensional space) and are therefore more useful for decoding overall than others which have only small readout weights.

We can also rank neurons according to the angle of their readout weight vector from the axis associated with one of the variables in the three-dimensional space. This quantifies the degree of specialization of the neuron for the chosen variable. We call the absolute value of the cosine of this angle the pure selectivity index. We can now ask whether neurons with a large pure selectivity index are particularly important for generalization, as quantified by the cross-condition generalization performance (CCGP). This can be tested by successively removing neurons with the largest pure selectivity indices from the data and performing the CCGP analysis on the remaining population of (increasingly) mixed selectivity neurons. The results of this ablation analysis are shown in Fig. S5, in which we plot the decay of the CCGP with the number of neurons removed. It demonstrates that while pure selectivity neurons are important for generalization (as expected in a pseudo-simultaneous population of neurons with re-sampled trial-to-trial variability, in which the principal axes of the noise clouds are aligned with the neural axes), they are not more important than neurons with overall large decoding weights which typically have mixed selectivity.

We can perform the same analyses also for the neural network model of Fig. 5. We focus on the neural representations obtained from the second hidden layer of the trained network. Fig. S6 shows a scatter plot of the decoding weights for the parity and magnitude variables. As in the neural recordings, many neurons exhibit mixed selectivity. We can again successively ablate neurons with the largest pure selectivity indices. Unlike for the case of the neural data, this only has a small effect on the cross-conditions generalization performance (see Fig. S6). Presumably, this is because here the neural activities are simultaneously 'recorded', and therefore the decoder can use information contained in the correlations between different neurons in these representations.

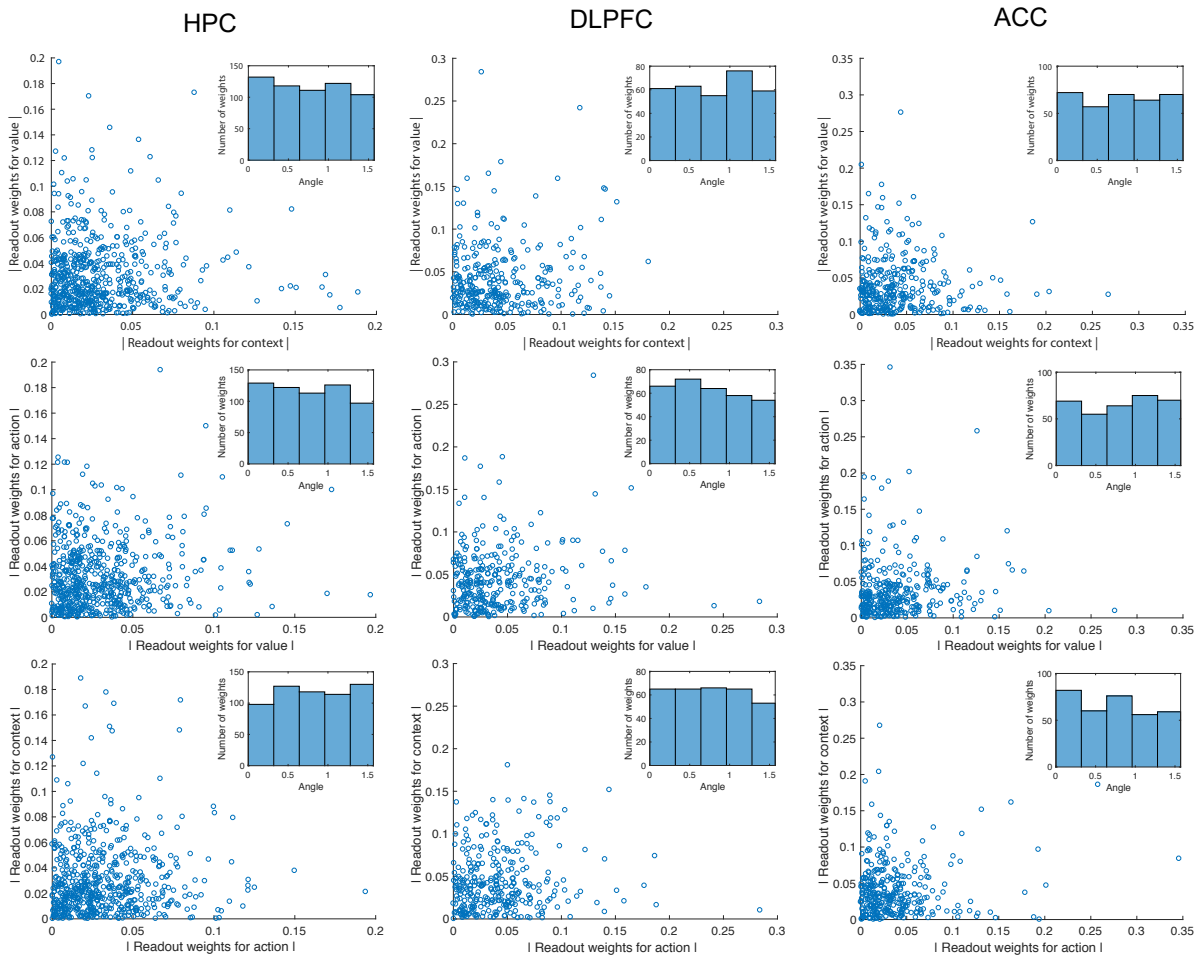


Figure S4: Two-dimensional scatter plots of the absolute values of the (normalized) decoding weights for the three task-relevant variables. The three columns (from left to right) correspond to HPC, DLPFC, and ACC. The three rows show the magnitudes of the weights for pairs of variables plotted against each other: context vs. value (top), value vs. action (middle), and action vs. context (bottom). The inset in each scatter plot shows a histogram of the weight counts as a function of the angle from the vertical axis (in radians). These distributions are approximately uniform, and therefore pure selectivity neurons (whose weights would fall close to one of the axes in the scatter plots) are not prevalent. Similar distributions have been observed in the rodent hippocampus [46].

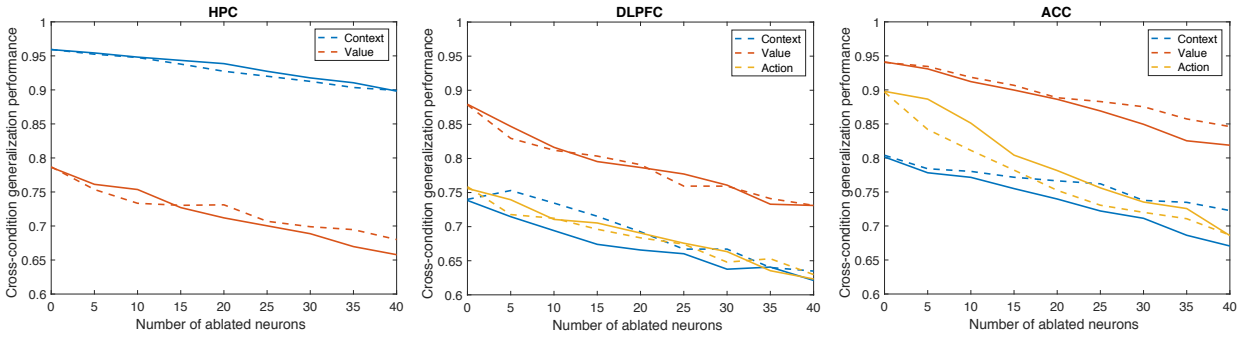


Figure S5: Cross-condition generalization performance as a function of the number of ablated neurons for the HPC (left), DLPFC (middle), and ACC (right). The solid lines show the decay of the CCGP if we successively remove the neurons with the largest pure selectivity indices for context (blue), value (red) or action (yellow). The dashed lines show the decline of the CCGP for the same three variables if we instead ablate neurons with the largest sum of squares of their three decoding weights (i.e., those with the radial position furthest from the origin in their three-dimensional weight space), independently of their pure selectivity indices. The two sets of curves are rather close to each other, and thus these two sets of ablated neurons are of similar importance for generalization. (For HPC, the CCGP of the action variable is always below chance level for both curves; not shown). This is similar to what has been observed in simulations of deep networks [47].

Unlike the case of resampled noise in a pseudo-simultaneous population, the principal axes of the noise clouds are not necessarily aligned with the neural axes. For such a simultaneously observed representation there is no reason why pure selectivity neurons should contribute preferentially to the generalization ability of the network.

S3 Deep neural network models of task performance

The study of performance-optimized neural networks is becoming increasingly common practice in computational neuroscience. This technique consists in training an artificial neural network model on the simulated version of a behavioral task of interest, and examining the neural activations generated by the neural network while or after learning the task. Following such practice, we train neural networks to perform a simulated version of our task under two learning paradigms and we show that after learning the neural representations are similar to those recorded in the experiment. In particular, we show that the learning paradigms that we use both lead to neural representations in which context is in an abstract format and multiple abstract variables are simultaneously represented.

As a first general learning paradigm we investigate *Reinforcement Learning*: our neural network model simulates the behavior of the animals engaged in the task and is trained by trial and error to maximize its performance on the simulated task. We then take a drastic simplification step and model the interaction between the animal and its environment during the task as a *super-*

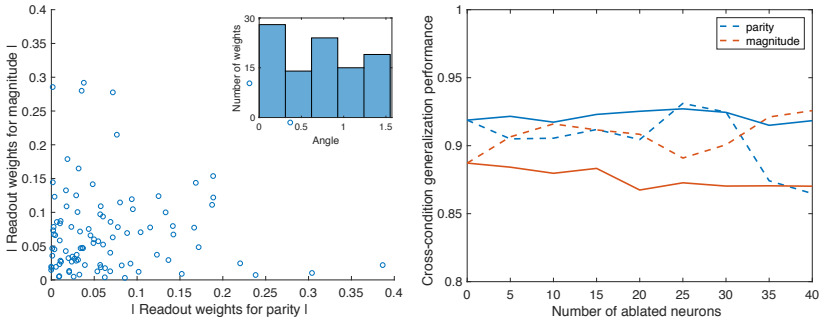


Figure S6: Specialization and ablation analyses of second hidden layer in the neural network of Fig. 5. Left: Two-dimensional scatter plot of the absolute values of the (normalized) decoding weights for the parity and magnitude dichotomies, as in Fig. S4. Right: CCGP when training on three digits from either side of the magnitude (red) and parity (blue) dichotomies and testing on the fourth one as a function of the number of ablated neurons. We ablate the neurons with the largest pure selectivity indices (solid lines), or the ones with the largest sum of squared decoding weights (dashed lines), as in Fig. S5. Note that even though we dont z-score the neural representation for the computations of the CCGP, or for any of the analyses shown in Fig. 5, the decoding weights shown here (and used to select the ablated neurons) are taken from classifiers trained on z-scored representation, since otherwise they would hardly be indicative of the relative importance of different neurons for decoding.

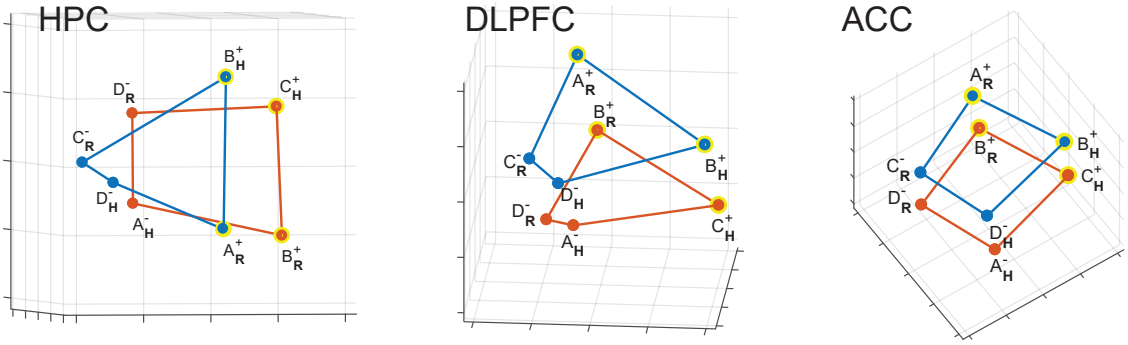


Figure S7: Representation of action of the previous trial in the three brain areas. The MDS plots of Figure 3 are rotated to highlight the dependence on the action of the previous trial (H=Hold, R=Release; in boldface). In DLPFC and ACC the points corresponding to the conditions with the same action are on the same side of the plot. In these areas, action is decodable and in an abstract format. In HPC all the 8 points are distinct, action is decodable, but it is not in an abstract format. Indeed, the H and R points are along two almost orthogonal diagonals. Notice that value (+/− symbols) and context (red/blue; less clear from this perspective) are in an abstract format in all three brain areas.

vised learning paradigm: our network model is essentially trained to output the correct (assumed to be known) behavioral response and predict the resulting outcome, given the current stimulus, and the sequence of stimulus, correct response and outcome in the previous trial. As we will see, this simplification still preserves the main features of the activity patterns generated by the more complex Reinforcement Learning model, while at the same time providing us with direct control of the statistics of the inputs presented at training (which are no longer dictated by the behavior of the model itself), thereby enabling us to systematically study their effect on the abstraction of the variables emerging during learning of the task.

Reinforcement Learning model. We simulate a sequential version of the task that is used as an environment to train by Reinforcement Learning a decision policy (an agent) parametrized as a multi-layer network with two hidden layers (whose size is indicated in Figure S8a), three outputs (one for each of the actions available to the agent), and ReLU activations.

All observables in the environment are encoded and presented to the network as binary input vectors with added Gaussian noise sampled independently for every trial. Figure S8a details the construction of these input vectors. At the beginning of a simulation we randomly sample 32-dimensional binary vectors encoding each of the values of all features appearing in the task (previous stimulus, previous action, previous outcome and current stimulus). An input representing a given task condition is then composed by stacking the binary vectors corresponding to the features describing the condition, and adding centered Gaussian noise with amplitude of 0.4.

An episode in the environment consists of two steps. In a first step corresponding to the pre-stimulus interval in the task, the agent observes the stimulus, action and outcome of the previous trial. The correct action at this step is "Wait", upon which the agent receives a reward of 0. If the agent mistakenly selects "Hold" or "Release", it receives a negative reward of -1 and the episode is terminated. In the second step of each episode, in addition to the observables presented in the first step, the agent observes the stimulus of the current trial (Figure S8a). At this point the agent has to select the correct action between "Hold" and "Release", which depends on the currently active context and the current stimulus identity. Upon selection of the correct action, the agent receives a reward of +1 for a rewarded trial, or a reward of 0 in the case of an unrewarded trial. Again, if the agent selects the wrong action, it receives a negative reward of -1. At this point the episode terminates, and a new one is initiated whose context is chosen to be identical to the one of the previous trial with a probability of 98%. Notice that information about the context identity is never made apparent in the observations, but is only reflected in the determination of the correct action, given the current stimulus. As a consequence, assuming that the context hasn't changed between the previous and current episode, the current context can be inferred from the triplet of previous stimulus, previous action and previous outcome, since these uniquely determine the correct action in the previous episode and therefore its context.

The particular choice of dividing each episode into two steps overlapping two adjacent trials was dictated by the requirement of being able to analyze a trial epoch that would correspond to the pre-stimulus interval that was analyzed in the recorded data. Moreover, explicitly providing information about the previous trial in the input to the network allows us to eschew complicated

Figure S8: Reinforcement Learning (RL) model trained with Deep Q-learning (DQN) on a simulated version of the serial reversal-learning task. a., b. An episode in the simulated environment is composed of two sequential steps, each corresponding to one trial. In the first step, corresponding to the pre-stimulus interval in the trial, the network receives as observations the stimulus, the action and the outcome in the previous trial, at which point it is required to issue the action "Wait" to initiate a new trial. In the second step of the episode the network receives the current stimulus (in addition to all observations presented in the first step) and is required to issue the correct action in the task ("Hold" or "Release"). If at any step the agent issues an incorrect action, it receives a reward of -1 and the episode terminates. Otherwise, it receives a reward of 0 after correctly issuing "Wait" at the end of the first step, and it receive a reward corresponding to the value of the stimulus in the task at the end of the second step (+1 or 0 depending upon whether it is a rewarded trial type). c. The agent is parametrized as a two-hidden layers neural network trained with DQN. We add a reconstruction loss to the optimization loss (indicated in blue in the lower right corner of the panel) that forces the network to find representations in the L2 hidden layer that can linearly reconstruct the inputs (i.e., the reconstruction loss is the mean squared error between the inputs and a linear transformation of the L2 activations). d. Learning curves for the DQN model on the serial-reversal task. The blue line is the fraction of correct actions averaged over 100 random initializations of the network and blocks of 50 episodes. The gray lines show the performance of 10 randomly chosen individual networks (also in blocks of 50 episodes).

mechanisms to preserve information across time steps, such as for example having to resort to a recurrent neural network model. Future studies will consider these more complicated mechanisms. The environment modeling choices that we just illustrated are therefore instrumental to allowing us to model the agent with the very simple feed-forward architecture described in Figure S8a. The only slightly unconventional element in our policy network architecture is the addition of a *reconstruction loss* (Figure S8a, lower right). This consists in a weighted term added to the optimization cost that is proportional to the squared difference between the input to the networks and a linear transformation of the activations of the second hidden layer. The role of this additional cost is to allow us to tune the amount of information that the hidden layer representations encode about the inputs. Practically speaking, this provides a mechanism to counteract a tendency that we observed in our simulations: the fact that larger numbers of training epochs and increasing depth tend to be associated with previous outcome and previous action variables of decreasing abstraction (as measured in terms of CCG performance and parallelism score). This effect will be more quantitatively analyzed by means of the supervised learning model in the next section.

The policy network described in the previous paragraph is trained to select the correct action with Deep Q-learning[16]. In particular, we use a frozen copy of the policy network that is being updated only every 10 episodes to estimate the Q-values corresponding to an observation-actions-reward triplet randomly sampled from a replay memory buffer of 5000 steps in the environment. After every step a plastic copy of the policy network is being trained to fit the Q-values estimated from the frozen policy network by using one step of adam-SGD[48] (with the default parameters in PyTorch [49], aside from a learning rate of 0.005 with a decay factor of 0.999) and mini-batch of size 100. The use of a slowly updated frozen copy of the policy network to estimate the Q-values is a technique that had been proposed in [16], and helps stabilizing the learning procedure. Moreover, as customary to promote exploration in Reinforcement Learning, actions are selected using an ϵ-greedy algorithm, i.e., they are selected based on the largest Q-value with a probability $1 - \epsilon$, and selected uniformly at random with probability ϵ. We set $\epsilon = 0.9$ at the beginning of learning and decrease it exponentially to 0.025 with a time constant of 500 steps. Finally, we use a discount factor of $\gamma = 0.99$.

Figure S8d shows the average learning curve for a population of 100 policy networks initialized at random and trained on random instantiations of the environment described in the previous paragraphs. The blue curve shows the fraction of correct terminal actions within blocks of 50 episodes, averaged across 100 networks, as a function of episode.

The results of analyzing the activity of the Reinforcement Learning model are presented in Figure S9. All analyses are conducted during the first step in the simulated tasks, which emulates the pre-stimulus interval in the experimental task, on a population of 100 randomly initialized and independently trained networks. The figure shows the results of the analyses averaged over the population of models, both before (grey data points) and after training (orange data points). We compute decoding accuracy (first row of panels), parallelism score (second row) and CCG performance (third row) for context (first column), value (second column) and action (third column).

Focusing on the first column of Figure S9 we see that the *context* cannot be decoded with above-chance accuracy (50%) in the input but can already be decoded with high accuracy starting from the randomly initialized first hidden layer (gray datapoints). This is consistent with previous

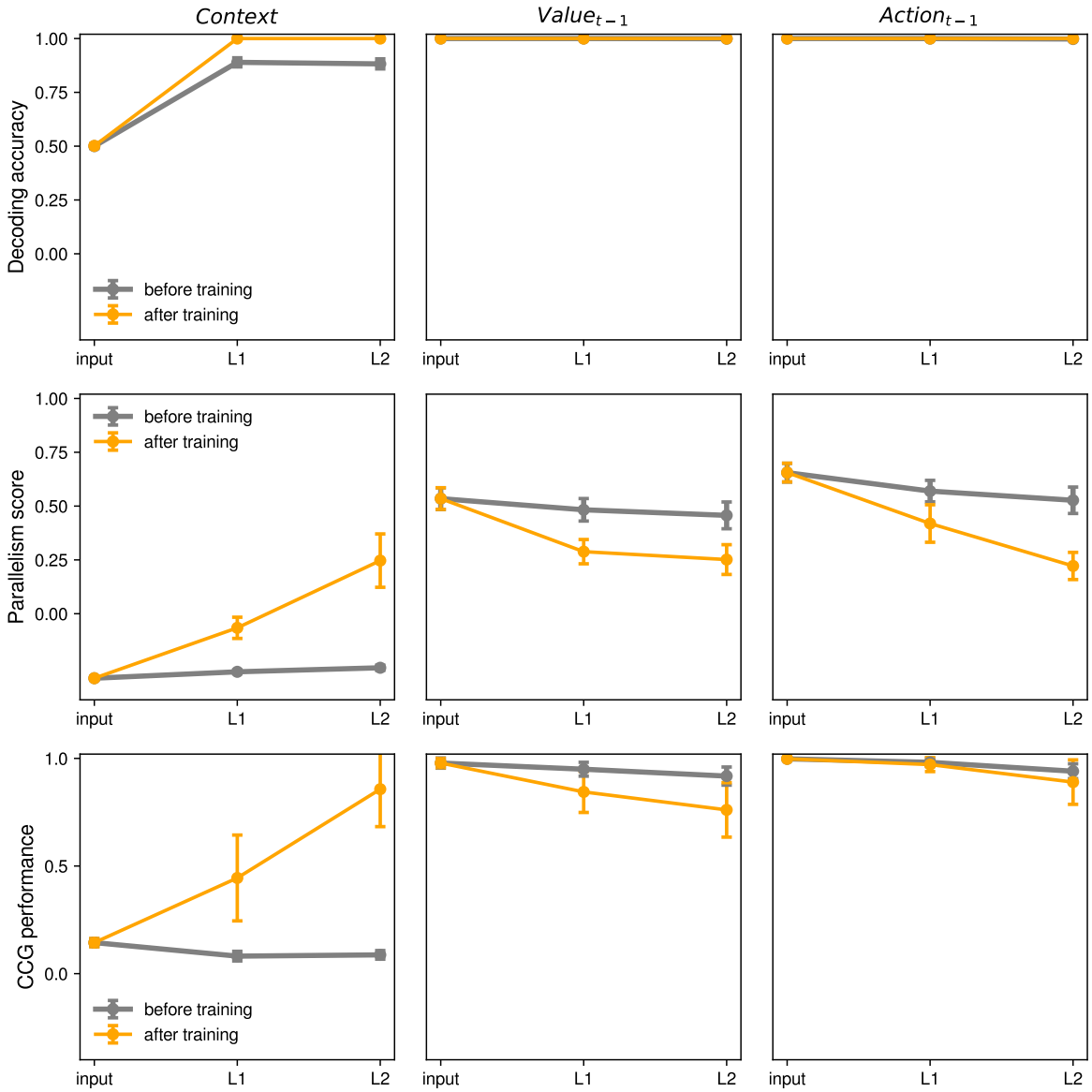


Figure S9: Analysis of the activity generated by the Reinforcement Learning model: *decoding accuracy* (first row), *parallelism score* (second row) and *cross-condition generalization (CCG) performance* (third row) during the simulated first step (pre-stimulus epoch) in the task, for the variables *context* (first column), *previous value* (second column) and *previous action* (third column). In each panel we plot one of the quantities of interested as a function of the layer where it is measured in the architecture: inputs, first hidden layer (L1) and second hidden layer (L2) (see Fig. S8). The plots show the mean quantity of interest across 100 randomly initialized and independently trained networks, before training (grey data points) and after training (orange data points). Error bars indicate standard deviations computed over the same distribution of 100 networks.

results that show that large enough networks of randomly connected neurons with non-linear activations generate linearly separable patterns with high-probability (see [27,50,51]).

From Figure S9, second row, we see that, despite being decodable with high accuracy in both hidden layers, context only becomes appreciably abstract in layer L2, where on average across the models population CCG performance becomes significantly higher than chance (i.e., 50%). Correspondingly, the average parallelism score in L2 is fairly high at around 0.25 (Figure S9, first plot in the second row). Moreover, in both hidden layers of the trained models, the parallelism score and CCG performance for context are significantly higher than in untrained models. In this respect, it is worth noting that for context the CCG performance of the randomly initialized models is consistently and considerably below chance level. This reflects the anti-correlation structure induced by the contextual character of the task: stimuli have to be associated to opposite context labels in the two contexts, which will naturally induce a linear decoder trained on a subset of conditions to predict the wrong context for the untrained conditions. This structure is also reflected in the parallelism score which tends to be negative for untrained models.

As for the previous value and action variables, they maintain relatively high CCG performance and parallelism score after training, but we notice a general tendency for representations to become progressively less abstract as a function of training and depth of the layer. This presumably reflects the fact that value and chosen action in the previous trial are not features that are directly predictive of the current value and action, although they can be utilized to compute the current context and therefore indirectly infer the current value and action.

Besides comparing the parallelism score and CCG performance of the trained models with those of the untrained models as baseline, we can quantify their statistical significance using the more stringent statistical tests developed to analyze the experimental data. In particular, we can use the geometric random model as a null-distribution to establish the statistical significance of CCG performances and use a shuffle test as a null-distribution for parallelism scores (as in Figure 4). Accordingly, Figure S10 shows beeswarm plots of the probability that the CCG performance and parallelism score of each of the trained Reinforcement Learning models is above the distributions given by the geometric random model and a shuffle test, respectively, for all variables in the task individually (context, previous value, and previous action) and all variables simultaneously for the same model (All). These probabilities are empirically estimated over 100 trained models and 1000 samples of the null-distributions per trained model. In Figure S10 we also display the mean of the null-distributions (dotted lines) and the 95th percentile (dashed line). Above each plot we report the fraction of trained models whose parallelism score and CCG performance is above the 95th percentile of the null-distribution. The main result is that, particularly in layer L2, a considerable fraction of trained models display a parallelism score and CCG performance that is above the 95th percentile of the null-distributions. For instance, in L2 40% of the trained models have a parallelism score that is higher than 95% of the shuffle null-distribution simultaneously for all variables, while 11% have a CCG performance above 95% of the null-distribution obtained with the random model.

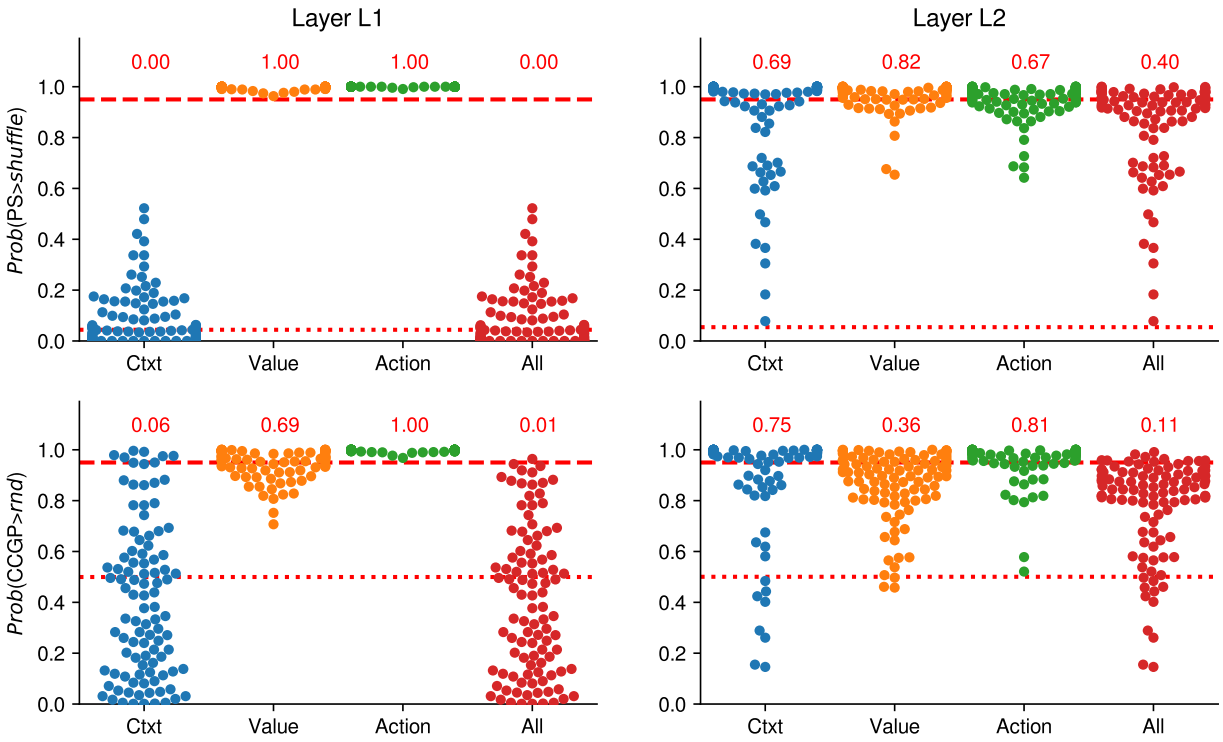


Figure S10: Beeswarm plots of the probability of Parallelism Score and CCG performance of trained Q-learning models to be above the null-distributions given by the geometric random model and a shuffle test of the data (empirically estimated with 1000 samples per model), respectively (see Figure 4). Every dot in the graph represents a trained model. The first row of plots shows the probability that the Parallelism Score of each model for every variable individually (Context, Value and Action) and all variables simultaneously (All) is above the null-distribution given by a shuffle of the data in layer L1 (first column) and L2 (second column). The second row is laid out similarly as the first one, but shows the probability that the CCG performance of each model is above the null-distribution given by the geometric random model. The dotted lines correspond to the mean of the null-distribution, and the dashed lines correspond to the 95th percentile of the null-distribution. The numbers written in red above each plot report the fraction of models that are above the 95th percentile threshold.

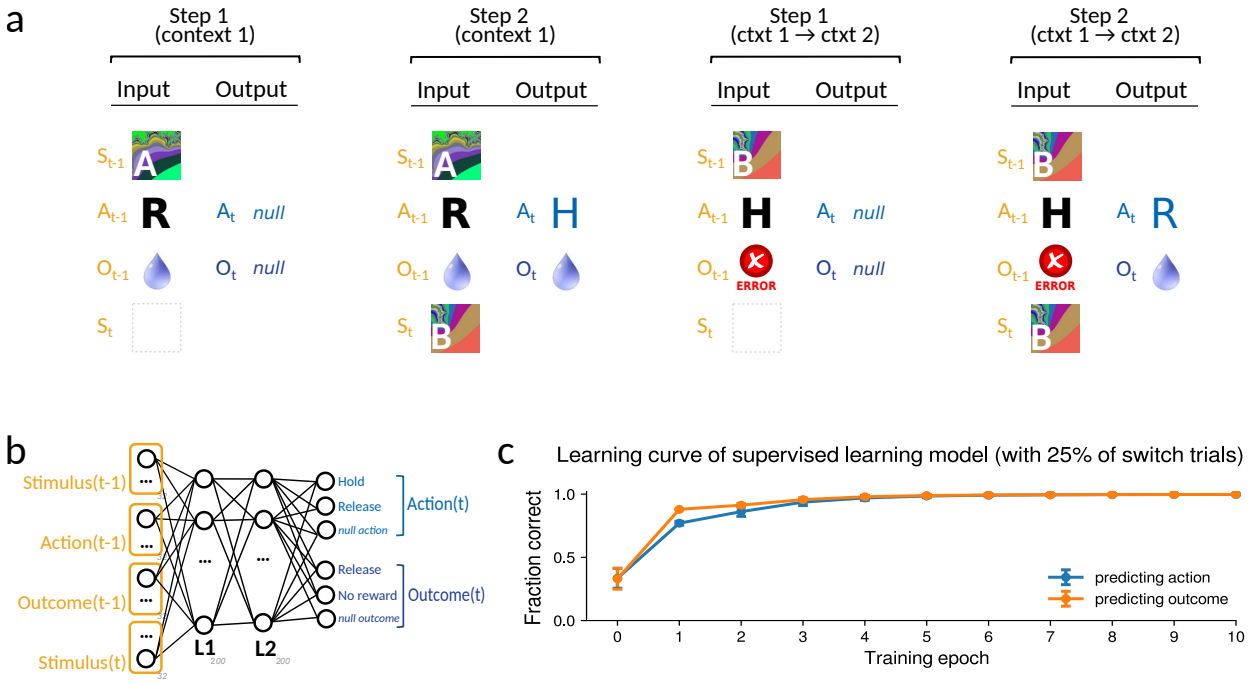


Figure S11: Supervised learning model. a. The supervised learning model is trained on two types of input-output combinations, corresponding to sequential steps in a trial. The inputs for the first step (the pre-stimulus interval in the trial) are composed of the stimulus, the action and the outcome (reward) in the previous trial, while the target output is a null action and value (corresponding to a third null action and a third null outcome neuron). Inputs corresponding to the second step encode the current stimulus (along with the features that were already presented in the first step), while the output encodes the correct action (Hold or Release) and outcome (Reward or No reward) in the trial. Besides being distinguished by whether they correspond to the first step or second step in the trial, generated inputs can be distinguished depending on whether or not they correspond to a switch trial. Inputs generated from non-switch trials (see first and second input-output combinations in the panel) define the correct action and subsequent outcome, under the assumption that the current context is the same as in the previous trial (encoded in the input). Inputs generated from switch trials are characterized by outcome features encoding an error in the previous trial (see second and third input-output combinations in the panel), implying that the observed stimulus-action combination is incorrect and counterfactually defining the currently correct context and action. b. The neural network is a multi-layer network with two hidden layers (number of unites indicated in the figure) trained with backprop in a supervised way to output the correct action and outcome. c. Learning curves for the (simultaneously learned) action and value prediction tasks at the end of the second step (corresponding to the response and value prediction of the animal) as a function of training episodes. Every epoch consists of 128 mini-batches of 100 noisy versions of the inputs just described. Data points are the means and standard deviations across 100 distinct random initializations of the network.

Supervised Learning neural network model. Besides investigating a Reinforcement Learning model of the task, we also devised simulations in a much simpler open-loop learning paradigm: supervised learning. In this case, the network is trained to associate inputs corresponding to the stimuli presented to the monkeys with outputs corresponding to the correct action and expected outcome. Given a task condition, the inputs are generated in the same way as for the Reinforcement Learning model. At the beginning of a simulation we randomly sample 32-dimensional binary vectors encoding each of the values of all features appearing in the task (previous stimulus, previous action, previous outcome, current stimulus, correct action and predicted outcome). An input representing a given task condition is composed by stacking the binary vectors corresponding to the features describing the condition, and adding centered Gaussian noise with amplitude of 0.4. A corresponding output is built by stacking the binary vectors encoding the correct action and predicted outcome in the current condition (see Figure S11). As for the Reinforcement Learning case, we simulate two task epochs for each trial type: a pre-stimulus epoch defined by the previous outcome, previous action and previous stimulus (the activity of the input neurons encoding the current stimulus is set to zero), and a second epoch where also the current stimulus is presented. For inputs corresponding to a pre-stimulus epoch the network is trained to output a 'null' action by activating a third action neuron, and a 'null' outcome by activating a third outcome neuron.

Training proceeds in epochs of 128 mini-batches of input-output pairs generated from the first step in the trial (where inputs are the stimulus, action, and outcome in the previous trial, while outputs are the null action and outcome) and the second step in the trial (where inputs are the same as in the first step with in addition the stimulus in the current trial, while the outputs are the correct action and predicted output), randomly intermixed. A fraction of 0.25 inputs-output pairs are generated so as to encode switch trials in the task, i.e. error trials that are following a contextual switch. The implicit assumption here is that the agent does not make mistakes, unless there is a contextual switch, which means that switch trials will be characterized by an outcome indicating an error (see Figure S11a, two rightmost input-output pairs). Errors are encoded with outcome neurons set to a fixed random binary pattern corresponding to a negative outcome in the previous trial, while the target output patterns are chosen so as to correspond to the opposite context compared to the one that was (incorrectly) followed in the previous step.

We train a population of 100 randomly initialized networks on the supervised learning version of the task. Figure S11b shows the resulting learning curves as a function of training epochs in terms of the average accuracy across the population at correctly predicting the null action and outcome after the first step, and correctly predicting the correct action and outcome after the second step, respectively (error bars denote the standard deviation of the accuracy across the population of trained networks).

The main results are analogous to what we observed for the Reinforcement Learning case. From Figure S12 we see that the *context* again cannot be decoded in the input but can already be decoded with high accuracy starting from the randomly initialized first hidden layer. Context becomes abstract in layer L2, where on average CCG performance (across the population of models) becomes significantly higher than chance. Correspondingly, the average parallelism score in L2 is fairly high, hovering around 0.25. In both hidden layers of the trained models context has a parallelism score and CCG performance that are significantly higher than in the untrained models.

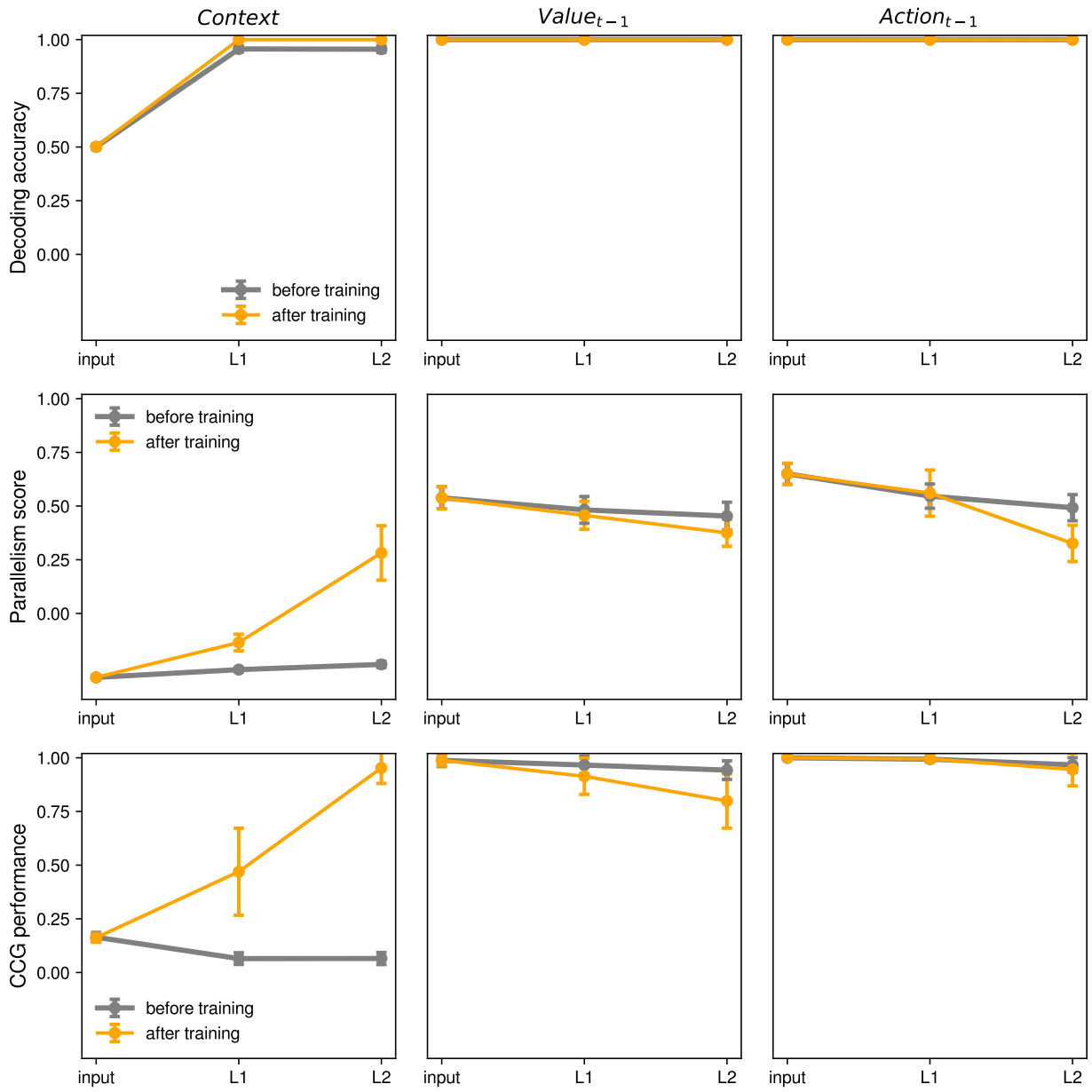


Figure S12: Supervised learning model, activity analysis: *decoding accuracy* (first row), *parallelism score* (second row) and *cross-condition generalization (CCG) performance* (third row) during the simulated first step (pre-stimulus epoch) in the task, for the variables *context* (first column), *previous outcome* (second column) and *previous action* (third column). In each panel we plot one of the quantities of interest as a function of the layer where it is measured in the architecture: inputs, first hidden layer (L1) and second hidden layer (L2). Each point in the plots represents the mean quantity of interest across 100 randomly initialized and independently trained networks, before training (grey data points) and after training (orange data points). Error bars indicate standard deviations computed over the same distribution of 100 networks. The lines connect points computed within the same training condition (before training or after training) in adjacent layers in the architecture.

After training the previous outcome and action variables maintain CCG performances and parallelism scores that are very close to the untrained models. However, as we noticed earlier, we note a slight decrease in abstraction of value and action that seems to correlate with an increase in the abstraction of context. We will study this effect in Figure S15.

As in the case of the Reinforcement Learning model, we compare the Parallelism Score and CCG performance of the trained models to the null-distributions obtained from a shuffle of the data and the geometric random model, respectively. Figure S13 shows beeswarm plots of the probability that the CCG performance and parallelism score of each of the trained supervised learning models is above the distributions given by the geometric random model and a shuffle test, respectively, for all variables in the task individually (context, previous value, and previous action) and all variables simultaneously for the same model (All). These probabilities are empirically estimated over 100 trained models and 1000 samples of the null-distributions per trained model. Figure S13 is laid out similarly to Figure S10. As in the Reinforcement Learning case a considerable fraction of trained models display parallelism score and CCG performance that is above the 95th percentile of the null-distributions, particularly in hidden layer L2. For instance, in L2 91% of the trained models have parallelism score that is higher than 95% of the shuffle null-distribution simultaneously for all three variables, while 40% have CCG performance above 95% of the null-distribution obtain with the random model.

Next we set out to study the effect that we noticed whereby abstraction of value seems to be negatively correlated with abstraction of context. First, we hypothesize that abstraction of value might be positively correlated with the fraction of switch trials, since maintaining information about the previous outcome (i.e., the value) is crucial to identify a contextual switch. Figure S14 shows scatter plots of the parallelism score and CCG performance for value in the hidden layer L2 of 100 supervised learning models trained as in the previous paragraphs but with different fractions of switch trials (between 0 and 0.5), revealing that indeed both quantities are positively correlated with the fraction of switch trials. Next we utilize the fraction of switch trials as an independent variable to control parallelism score and CCG performance of value, and quantify the correlation between abstraction of value and context. This is done in Figure S15. The first panel of Figure S15 shows a statistically significant negative correlation -0.34 between the parallelism score for value and the parallelism score for context in hidden layer L2 of the supervised learning model. Analogously, the second panel of Figure S15 shows that the logit transform of the CCG performance for context is also negatively correlated with the CCG performance of value (we used the logit transform, because CCG performance for context tends to saturate close to one for low values of CCG performance for value). These results quantify the effect that we had qualitatively alluded to earlier in the Reinforcement Learning model that caused abstraction of value to decrease as abstraction of context increased, and which prompted us to introduce the reconstruction loss term.

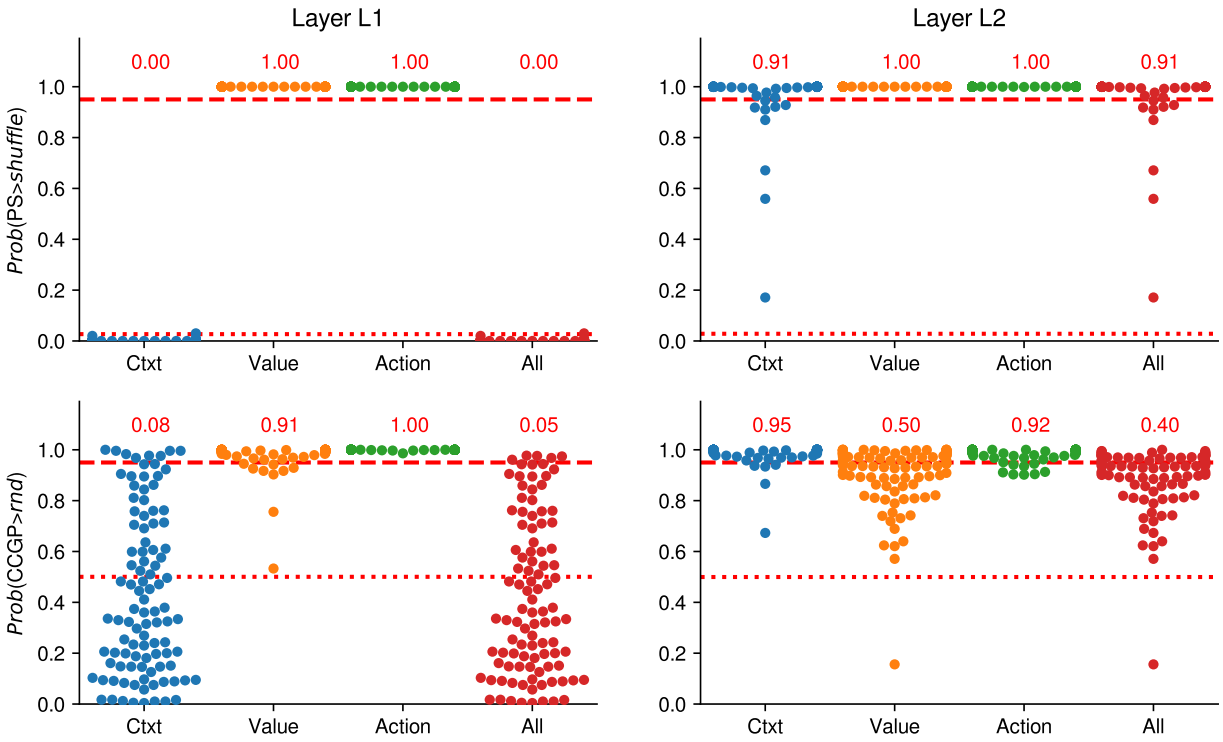


Figure S13: Beeswarm plots of the probability of Parallelism Score and CCG performance of trained supervised learning models to be above the null-distributions given by the geometric random model and a shuffle test of the data (empirically estimated with 1000 samples per model), respectively (see Figure 4). Every dot in the graph represents a trained model. The first row of plots shows the probability that the Parallelism Score of each model for every variable individually (Context, Value and Action) and all variables simultaneously (All) is above the null-distribution given by a shuffle of the data in layer L1 (first column) and L2 (second column). The second row is laid out similarly as the first one, but shows the probability that the CCG performance of each model is above the null-distribution given by the geometric random model. The dotted lines corresponds to the mean of the null-distribution, and the dashed line corresponds to the 95th percentile of the null-distribution. The numbers written in red above each plot report the fraction of models that are above the 95th percentile threshold.

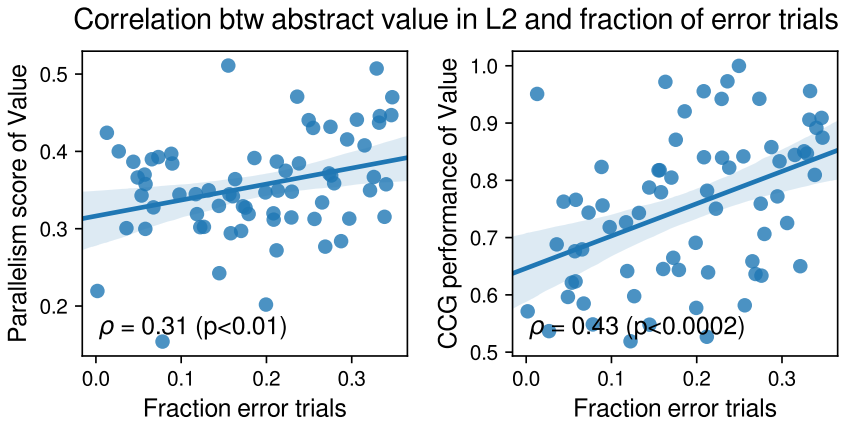


Figure S14: Correlation between abstraction of value and fraction of switch trials in hidden layer L2 of the supervised learning model. Both parallelism score and CCG performance for value are positively correlated with the fraction of switch trials with statistically significant Pearson correlation coefficients ρ (t-test).

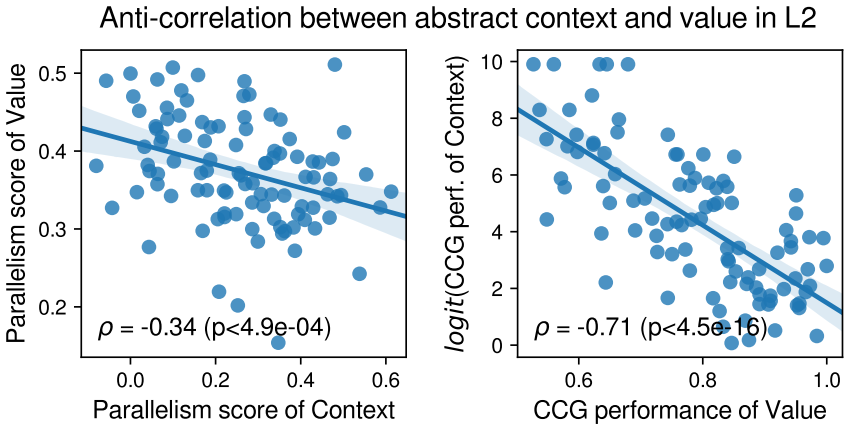


Figure S15: Anti-correlation between abstraction of value and abstraction of context in hidden layer L2 of the supervised learning model. Both parallelism score and CCG performance for value are negatively correlated with respectively the parallelism score and CCG performance for context. Since CCG performance for context tends to saturate close to one for low CCG performance for value, we applied a logit transform to the CCG performance for context, before computing the Pearson correlation coefficients.

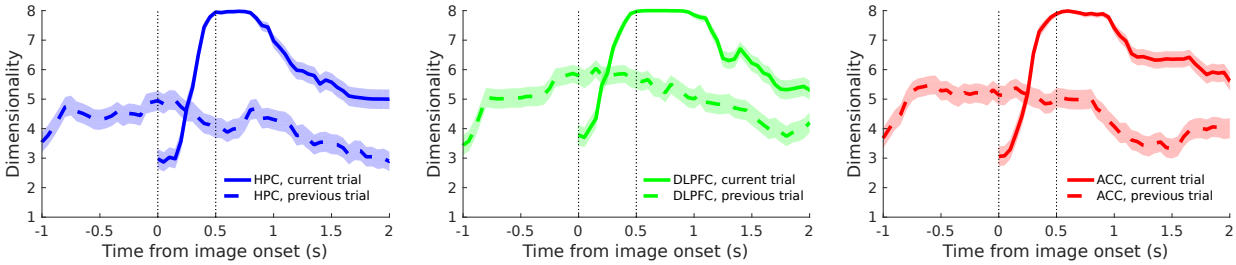


Figure S16: Dimensionality of the average firing rate activity patterns as a function of time throughout the trial. The left panel illustrates the result of the analysis developed in [52] on HPC, the central panel refers to DLPFC, and the right panel to ACC. Continuous lines refer to the analysis carried out on average firing rate patterns obtained by averaging spike counts according to the task conditions of the trial that was being recorded (current trial), while for dashed line we do the same but for conditions defined in the previous trial. The lines indicate the number of principal firing rate components that are larger than all noise components, averaged over 1000 re-samplings of the noise covariance matrix (see [52]). The shadings indicate the 95% confidence intervals estimated using the method for quantifying uncertainty around the mean of bounded random variables presented in [53].

S4 Dimensionality of the neural representations

We utilize a technique developed in [52] to estimate a lower bound for the dimensionality of the neural response vectors in a specific time bin during a task. Similarly to our other analyses, we build average firing rate patterns for all recorded neurons by averaging spike counts sorted according to task conditions indexing the trial where the activity is recorded (current trial) or the previous trial. The spike counts are z-scored and averaged in 500 ms time bins displaced by 50 ms throughout the trial. We then apply the method presented in [52] on the obtained average firing rate activity patterns independently within each 500 ms time bin. This procedure allows us to bound the number of linear components of the average firing rate patterns that are due to finite sampling of the noise, therefore providing an estimate of their dimensionality.

Figure S16 shows the result of this analysis for all neurons recorded in HPC, DLPFC and ACC for which we had at least 15 trials per condition. As we can see, for average firing rate patterns obtained by sorting spike counts according to the 8 conditions of the current trial (continuous lines), dimensionality peaks at its maximum possible value shortly after the presentation of the image for all three areas. The dimensionality for firing rate patterns obtained by sorting the activity according to the condition of the previous trial remains around 5 throughout the trial, which is close to the value to which dimensionality in the current trial decays towards the end of the trial.

S5 High dimensionality versus generalization: Flexible computation and abstraction are not mutually exclusive

The class of neural geometries we propose would require neurons to (at least approximately) exhibit linear mixed selectivity [13,14], which entails that neural representations have low dimensionality. In fact, the dimensionality of such a geometry would be equal to the number of the (in our case binary) abstract variables involved (not counting the possible offset from the origin of the neural activity space). This dimensionality is small compared to the maximal dimensionality, which grows exponentially with the number of variables, and would be equal to the total number of conditions minus one. In the idealized case in which all abstract variables have a parallelism score of one, the representations of the different conditions coincide with the vertices of a parallelotope, and it is easy to see that the dimensionality equals the number of variables, which is much lower than the maximal dimensionality.

It has been argued [13,14] that high-dimensional neural representations are often important for flexible computation, because a downstream area may in principle have to read out an arbitrary dichotomy of the different conditions (i.e., an arbitrary binary function) to solve different tasks. This can be achieved using simple linear classifiers (used to model the type of computation that a readout neuron may be able to implement directly) only if the dimensionality of the neural representation is maximal. This desire for flexible computations afforded by high dimensionality seems to be in direct opposition to the low dimensionality implied by the abstract neural geometries that allow for cross-condition generalization.

However, in fact there is a large class of geometries that combine close to maximal dimensionality with excellent generalization properties for a number of abstract variables (which form a preferred subset of all dichotomies). Maximal dimensionality implies decodability by linear classifiers of almost all dichotomies. We will refer to this classifier-based measure of dimensionality as the 'shattering dimensionality'. This quantity is similar to the one introduced in [13], where it was measured to be maximal in neural representations in monkey pre-frontal cortex. The geometries with high dimensionality and excellent generalization don't have unit parallelism scores for the abstract variables (they are not exactly factorizable). A simple way to construct examples of such geometries is to start from a simple factorizable case, namely a cuboid, and then distort it to reduce the parallelism score. We will illustrate this in the case of eight conditions. In this case, there are at most three completely abstract variables, as in our experimental data. We can generate an artificial data set with the desired properties by arranging the eight conditions initially at the corners of a cube (with coordinates plus/minus one), embedding this cube in a high-dimensional space by padding their coordinate vectors with zeros (here we use $N = 100$ dimensions, so we append 97 zeros), and acting on them with a random (100-dimensional) rotation to introduce linear mixed selectivity. We then distort the cube by moving the cluster center of each condition in a random direction - chosen independently and isotropically - by a fixed distance, which parameterizes the magnitude of the distortion. This operation reduces the parallelism score for the three initially perfectly abstract variables, which correspond to the three principal axes of the original cube, to values less than one. We sample an artificial data set of 1,000 data points per condition by assuming an isotropic Gaussian noise distribution around each cluster center.

On this data set we can run the same analyses as on our experimental data. In particular, we compute the parallelism score and the cross-condition generalization performance averaged over the three preferred (potentially abstract) dichotomies, with the results of these analyses shown in Fig. S17. In addition to these quantities, we compute across all 35 balanced dichotomies the average decoding performance (ADP), which is a quantity related to the shattering dimensionality. For small noise, the mean parallelism score of the abstract variables starts very close to one and from there decreases monotonically as a function of the magnitude of the distortion. The same is true for their mean CCGP, but its decline is much more gradual and almost imperceptible for small distortions. This means that for intermediate values of the displacement magnitude (of order one), we still see excellent generalization properties, and the three preferred variables are still in an abstract format.

In contrast, the ADP increases as a function of the magnitude of the distortion, due to the increased dimensionality of the representation. Balanced dichotomies include the most difficult (least linearly separable) binary functions of a given set of conditions, and if all of them can be decoded by linear classifiers the dimensionality of the neural representation will be maximal. For small values of the displacement magnitude (for which the neural geometry is three-dimensional) some dichotomies are clearly not linearly separable, and therefore the average decoding performance starts out at a value less than one (around 75%), but it steadily increases with the degree of distortion of the cube. Crucially, it reaches its plateau close to one before the CCGP drops substantially below one, i.e., there is a parameter regime in which this type of neural geometry exhibits almost maximal dimensionality enabling flexible computation, but at the same time also abstraction (in the form of excellent generalization properties) for the three preferred variables. Therefore, we can conclude that these two favorable properties are not mutually exclusive.

In the case of larger noise, the PS and CCGP start out at values substantially smaller than one already for zero distortion. The qualitative trends of all the quantities discussed remain the same, but the tradeoff between the average decoding performance and cross-condition generalization (i.e., of flexible computation and abstraction) is much more gradual under these circumstances.

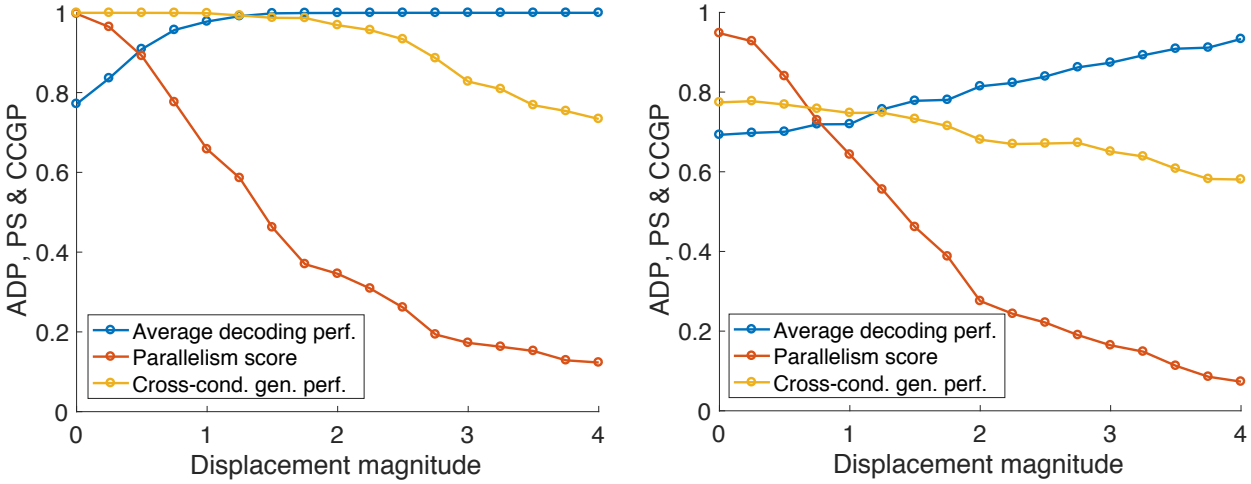


Figure S17: Analysis of an artificial data set generated by randomly embedding a (three-dimensional) cube in a 100-dimensional space, displacing its corners in independent random directions by a certain distance (displacement magnitude), and then sampling data points corresponding to the eight experimental conditions from isotropic Gaussian distributions around the cluster centers obtained from this distortion procedure. We plot the average decoding performance (ADP) across all balanced dichotomies, as well as the mean PS and CCGP of the three potentially abstract variables for low (left) and high noise (right), with noise sampled as i.i.d. unit Gaussian vectors multiplied by overall coefficients 0.2 and 1.0, respectively. For low noise, both the average decoding performance and the cross-condition generalization performance can be simultaneously close to one, indicating maximal dimensionality and the presence of three abstract variables for these representations. In the case of high noise, we observe a smooth tradeoff between the CCGP (abstraction) and the ADP (dimensionality).