nature neuroscience

ARTICLES

https://doi.org/10.1038/s41593-018-0310-2

Task representations in neural networks trained to perform many cognitive tasks

Guangyu Robert Yang[1,2], Madhura R. Joglekar[1,6], H. Francis Song[1,7], William T. Newsome[3,4] and Xiao-Jing Wang[1,5*]

The brain has the ability to flexibly perform many tasks, but the underlying mechanism cannot be elucidated in traditional experimental and modeling studies designed for one task at a time. Here, we trained single network models to perform 20 cognitive tasks that depend on working memory, decision making, categorization, and inhibitory control. We found that after training, recurrent units can develop into clusters that are functionally specialized for different cognitive processes, and we introduce a simple yet effective measure to quantify relationships between single-unit neural representations of tasks. Learning often gives rise to compositionality of task representations, a critical feature for cognitive flexibility, whereby one task can be performed by recombining instructions for other tasks. Finally, networks developed mixed task selectivity similar to recorded prefrontal neurons after learning multiple tasks sequentially with a continual-learning technique. This work provides a computational platform to investigate neural representations of many cognitive tasks.

The prefrontal cortex is important for numerous cognitive functions[1–3], partly because of its central role in task representation[4–7]. Electrophysiological experiments using behaving animals have found prefrontal neurons that are either selective for different aspects of a given task[8,9] or functionally mixed[10,11]. Much less is known about functional specialization of task representations at the neuronal level. Imagine a single-neuron recording that could be carried out with animals switching between many different tasks. Is each task supported by a 'private' cluster of neurons, does each task involve every neuron in the network, or somewhere in between? If two tasks require a common underlying cognitive process, such as working memory or decision-making (DM), what would be the relationship between their neural representations? In other words, what would be the 'neural relationship' between this pair of tasks? Would the two tasks use a shared cluster of neurons?

Humans readily learn to perform many cognitive tasks in a short time. By following verbal instructions such as 'release the lever only if the second item is not the same as the first', humans can perform a novel task without any training at all[6]. A cognitive task is typically composed of elementary sensory, cognitive, and motor processes[5]. At the computational level, correctly performing a new task without training requires composing elementary processes that are already learned. This property, called 'compositionality', has been proposed as a fundamental principle underlying flexible cognitive control[12]. In a neuronal circuit equipped with a compositional code, a new task might be represented as the algebraic sum of representations of the underlying elementary processes. Indeed, human studies have suggested that the representation of complex cognitive tasks in the lateral prefrontal cortex could be compositional[6,13]. However, these tasks involved verbal instructions; it is unknown whether non-verbal tasks commonly used in animal physiological experiments also display compositionality and whether compositional task structures can emerge in relatively simple neural network models.

For a network capable of performing many tasks, should it be clustered? Should its representation be compositional? Conceptually, the answer to either question can be yes or no, independently (Fig. 1a). A randomly connected network can potentially solve multiple tasks by mixing sensory stimuli and rule inputs in a high-dimensional space. Such a network will have no clustering and show no compositionality across tasks. A network where different tasks are represented by completely non-overlapping populations will show clustering but no compositionality. A network can be both clustered and compositional if common cognitive processes across tasks are represented by distinct clusters of neurons. Finally, imagine linearly mixing neuronal activity of a clustered, compositional network. The resulting neural activity would still be compositional, but no longer clustered at the single-unit level.

Verifying these hypotheses remains difficult with conventional experimental and modeling approaches. Experiments with laboratory animals have so far been largely limited to a single task at a time; on the other hand, human imaging studies lack the spatial resolution to address questions at the single-neuron level. Therefore, the lack of neural recordings from animals performing many different tasks leaves unanswered important questions regarding how a single network represents and supports distinct tasks. In principle, these questions could be addressed in neural circuit models, but designing a single neural circuit model capable of multiple tasks is challenging and virtually nonexistent.

To explore potential solutions to these problems, we took the approach of training RNNs[11,14–19]. In this work, we trained single RNNs to perform 20 cognitive tasks. We found that after training all tasks simultaneously, the emerging task representations were organized in the form of clustering of recurrent units. Through a systematic examination, we identified the conditions under which clusters emerge. We found that a simple form of compositional task representation emerges from training in our network models. These networks can be instructed to perform certain tasks by combining

[1]Center for Neural Science, New York University, New York, NY, USA. [2]Mortimer B. Zuckerman Mind Brain Behavior Institute, Department of Neuroscience, Columbia University, New York, NY, USA. [3]Department of Neurobiology, Stanford University, Stanford, CA, USA. [4]Howard Hughes Medical Institute, Stanford University, Stanford, CA, USA. [5]Shanghai Research Center for Brain Science and Brain-Inspired Intelligence, Shanghai, China. [6]Present address: Courant Institute of Mathematical Sciences, New York University, New York, NY, USA. [7]Present address: DeepMind, London, UK. *e-mail: xjwang@nyu.edu

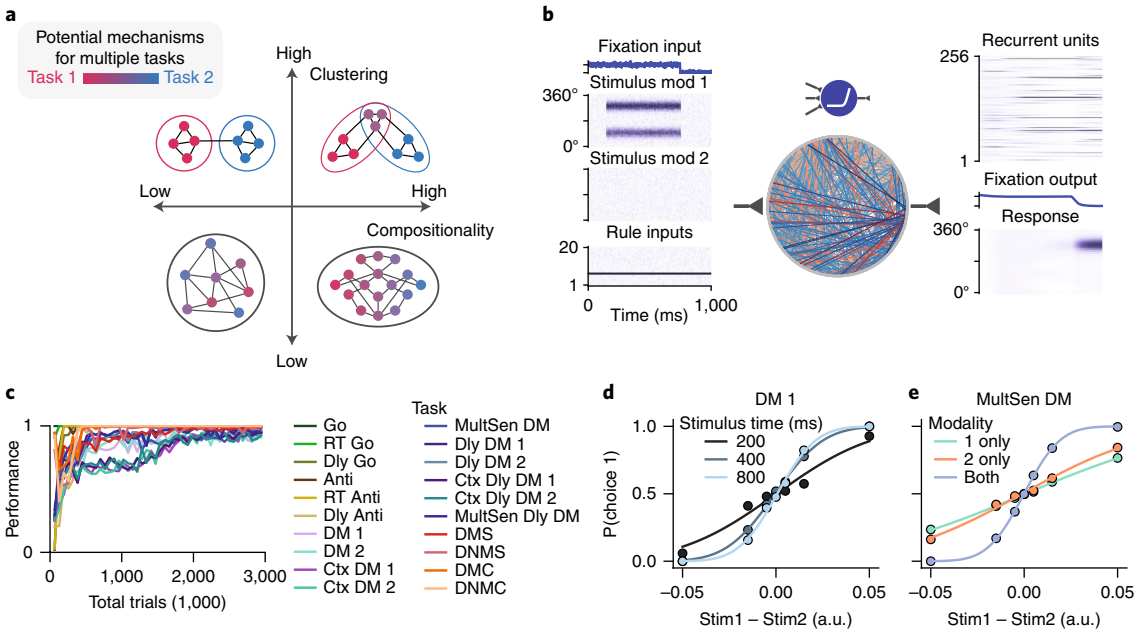

Fig. 1 | A recurrent neural network model is trained to perform a large number of cognitive tasks. a, Schematic showing how the same network can potentially solve two tasks with or without clustering and compositionality. **b**, An example of a fully connected recurrent neural network (RNN) (middle, 1% of connections shown) described by rate units receives inputs (left) encoding a fixation cue, stimuli from two modalities, and a rule signal (that instructs the system which task to perform in a given trial). The network has 256 recurrent units (top right) and it projects to a fixation output unit (which should be active when a motor response is unwarranted) and a population of units selective for response directions (right). All units in the reference recurrent network have non-negative firing rates. All connection weights and biases are modifiable by training using a supervised learning protocol. **c**, The network successfully learned to perform 20 tasks. **d**,**e**, Psychometric curves in two DM tasks. **d**, Perceptual DM relies on temporal integration of information, as the network performance improves when the noisy stimulus is presented for a longer time. a.u., arbitrary unit. **e**, In a multi-sensory integration task, the trained network combines information from two modalities to improve performance (compared with performance when information is only provided by a single modality). Ctx, context dependent; Dly, delayed; DMC, delayed match-to-category; DMS, delayed match-to-sample; DNMC, delayed non-match-to-category; DNMS, delayed non-match-to-sample.

instructions for other tasks. To mimic the process of adult animals learning laboratory tasks, we also trained networks to learn multiple tasks sequentially with the help of a continual-learning technique. The resulting neural representation in such networks can be markedly different from networks trained on all tasks simultaneously. Neural recordings from the prefrontal cortex of monkeys performing context-dependent DM tasks are consistent with the continual-learning networks. Our work provides a framework for investigating neural representations of task structures.

Results

Training neural networks for many cognitive tasks. To study how various cognitive tasks might be implemented in a single neural circuit, we first trained a RNN model (Fig. 1b) to perform 20 inter-related tasks. Most of these tasks are commonly used in neurophysiological studies of non-human animals and crucial to our understanding of the neural mechanisms of cognition. The chosen set of tasks includes variants of memory-guided response[20], simple perceptual DM[21], context-dependent DM[11,22], multi-sensory integration[23], parametric working memory[24], inhibitory control (for example, in anti-saccade)[25], delayed match-to-sample[26], and delayed match-to-category[27] tasks (Supplementary Table 1 and Supplementary Fig. 1).

We designed our network architecture to be general enough for all the tasks mentioned above, but otherwise as simple as possible to facilitate analysis. For every task, the network receives noisy inputs of three types: fixation, stimulus, and rule (Fig. 1b). The fixation input indicates whether the network should 'fixate' or respond (for example, 'saccade'). Thus, the decrease in the fixation input provides a 'go signal' to the network. The stimulus inputs consist of two modalities, each represented by a ring of input units that encodes a one-dimensional circular variable such as motion direction or color on a color wheel[18]. A single rule input unit is activated in each trial, instructing the network on which task it is currently supposed to perform. The network projects to a fixation output unit and a group of motor units encoding the response direction as a one-dimensional variable on a ring of outputs (for example, saccade direction, reach direction). All network units receive private noise. In the 'reference' setting of our networks, all units have non-negative and non-saturating activities to mimic biological neurons[28,29].

Before training, a network is incapable of performing any task. It is trained with supervised learning[11,15], which modifies all connection weights (input, recurrent and output) to minimize the difference between the network output and a desired (target) output. Notably, for the networks analyzed throughout most of the paper, all tasks were randomly interleaved during training. At the end, we will present results from sequential training of tasks. Below we show results obtained from networks of 256 recurrent units, and our results were robust with respect to the exact network size. After training, single network models achieved high behavioral performance across all tasks (Fig. 1c). Furthermore, by conducting a battery of psychometric tests, we demonstrated that the networks display behavioral features consistent with animal studies. In perceptual DM tasks, an example network achieved better performance with higher coherence and longer duration of the stimulus[21] (Fig. 1d and Supplementary Fig. 2a–f), and it combined information from different sources to form decisions[23] (Fig. 1e). In working memory tasks, the same network was able to maintain information

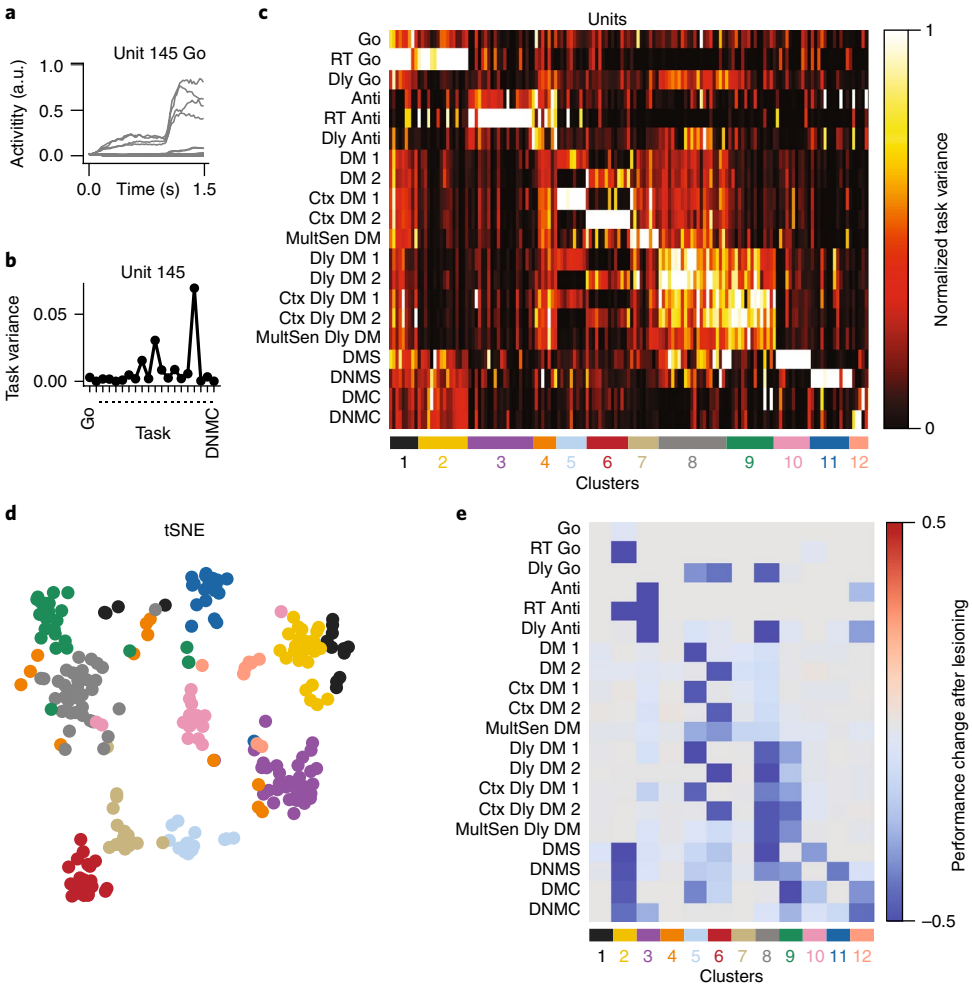

Fig. 2 | The emergence of functionally specialized clusters for task representation. a, Neural activity of a single unit during an example task. Different traces correspond to different stimulus conditions. **b**, Task variances across all tasks for the same unit. For each unit, task variance measures the variance of activities across all stimulus conditions. **c**, Task variances across all tasks and active units, normalized by the peak value across tasks for each unit. Units form distinct clusters identified using the *k*-means clustering method based on normalized task variances. Each cluster is specialized for a subset of tasks. A task can involve units from several clusters. Units are sorted by their cluster membership, indicated by colored lines at the bottom. **d**, Visualization of the task variance map. For each unit, task variances across tasks form a vector that is embedded in the two-dimensional space using *t*-distributed stochastic neighbor embedding (tSNE). Units are colored according to their cluster membership. **e**, Change in performance across all tasks when each cluster of units is lesioned.

throughout a delay period of up to 5 s[1,20,24] (50 times the single-unit time constant) (Supplementary Fig. 2g).

Functional clusters encode subsets of tasks in reference networks. The focus of our analysis was to examine the neural representation of tasks. After training, it is conceivable that each unit of the recurrent network is only selective in one or a few tasks, forming highly specialized task representations. On the other hand, task representations may be completely mixed, where all units are engaged in every task (Fig. 1a). We sought to assess where our reference networks lie on the continuum between these two extreme scenarios. In this section, we will focus our analyses on one example network.

To quantify single-unit task representation, we need a measure of task selectivity that is general enough that it applies to a broad range of tasks, and at the same time simple enough that it can be easily computed. We propose a measure that we call task variance (see Methods). For each task and each unit, the task variance computes the variance of that unit's noise-free response across conditions in that task (Fig. 2a). This measure quantifies the amount of stimulus information a unit conveyed during a task, without asking how that stimulus information is encoded. Units with different stimulus tuning can have the same task variance in a task. Task variance is agnostic about the task setup and can be easily computed in models and is also applicable to the analysis of experimental data.

By computing the task variance for all trained tasks, we were able to study how individual units are differentially selective in all of the tasks (Fig. 2b). For better comparison across units, we normalized the task variance of each unit such that the maximum normalized variance over all tasks was 1. By analyzing the patterns of normalized task variance for all active units, we found that units were self-organized into distinct clusters through learning (Fig. 2c,d and Supplementary Fig. 3a) (see Methods). We identified about 15 clusters in the network. The ideal number of clusters was chosen to maximize the ratio of intercluster to intracluster distances (Supplementary Fig. 4). Units belonging to the same cluster are mainly selective in the same subset of tasks. Units in the same cluster can have different incoming and outgoing connection weights however, simply as a result of different stimulus tuning (Supplementary Fig. 5).

To understand the causal role of these clusters, we lesioned each of them while monitoring the change in performance across all

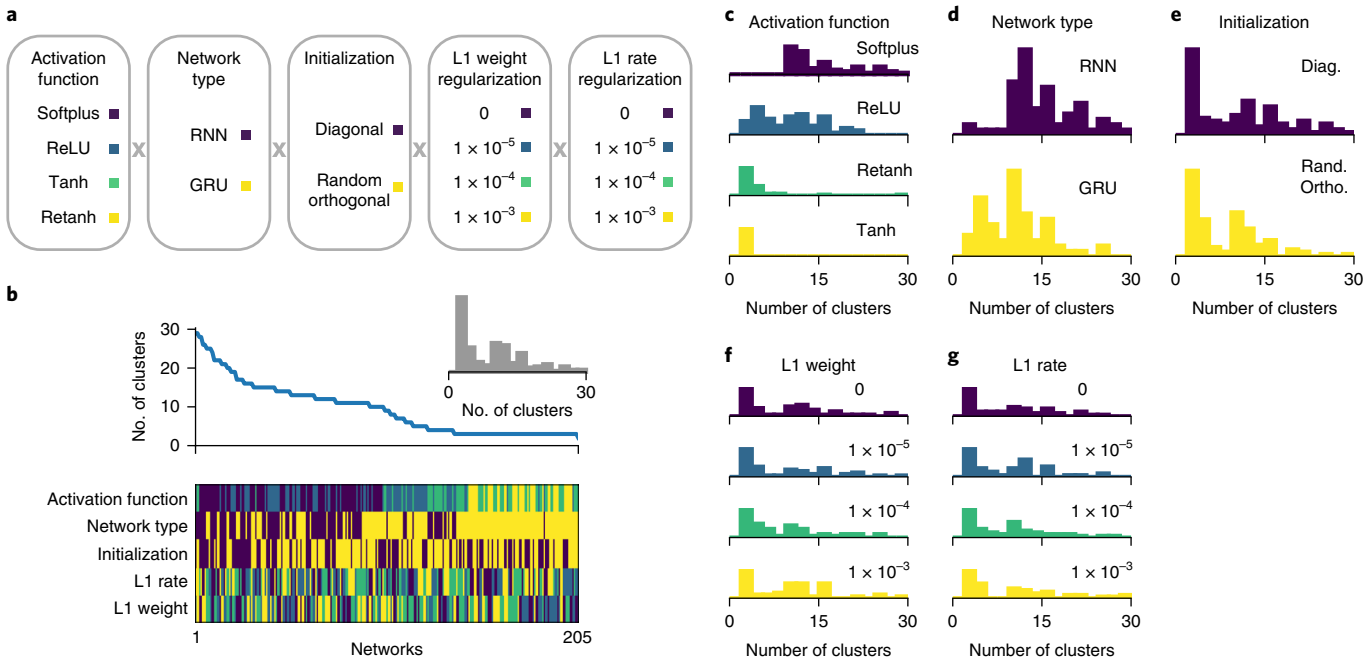

Fig. 3 | The activation function dictates whether clusters emerge in a network. a, A total of 256 networks were trained, each with a different set of hyperparameters. **b**, Top, the networks are sorted by their numbers of clusters. Bottom, the hyperparameters used for each network are indicated by colors, as defined in **a**. Inset, the distribution of cluster numbers across networks. Only networks that reached the minimum performance of 90% for every task are shown. **c**–**g**, Breaking down the number of clusters according to the activation function (**c**), network architecture (**d**), weight initialization (**e**), L1 weight regularization strength (**f**), and L1 rate regularization strength (**g**). In **e**–**g**, all networks as in **a** that learned all tasks are included. In **d**, only networks with Softplus and ReLU activation functions are shown, as no RNN with Tanh and ReTanh successfully learned all tasks.

20 tasks (Fig. 2e). We found one cluster (cluster number 3) that was specialized for the Anti-family tasks. Another two clusters (cluster numbers 5 and 6) were specialized for DM tasks involving modality 1 and 2, respectively. Furthermore, one cluster (cluster number 8) selective in the parametric working memory tasks (the delayed DM or Dly DM task family) was also selective in the perceptual DM tasks (the DM task family), indicating a common neural substrate for these two cognitive functions in our reference networks[30]. We can also study how units are clustered on the basis of epoch variance, a measure that quantifies how selective units are in each task epoch (Supplementary Fig. 3). One cluster of units presumably supports response generation, as it was highly selective in the response epoch, but not the stimulus epoch. These findings are robust across independently trained network with the same setting. Our results indicate that the reference networks successfully identified common sensory, cognitive, and motor processes underlying subsets of tasks, and, through training, developed units dedicated to the shared processes rather than to the individual tasks.

Assessing clustering across a wide range of models. We showed that networks trained to perform many cognitive tasks can develop clusters of units. Although connection weights in the network are adjusted with supervised learning, we specified the hyperparameters, such as the neuronal activation function (the input-output transfer function), the overall network architecture, and further training objectives. To understand how the emergence of clustering may depend on the hyperparameters used, we trained networks with four different activation functions (Softplus, rectified linear unit or rectified linear function (ReLU), hyperbolic tan function (Tanh), and rectified Tanh or ReTanh), two different architectures (leaky RNN and leaky gated recurrent unit (GRU) network), two weight initializations (diagonal and random orthogonal), four levels of L1 regularization on weights, and four levels of L1 regularization on activity (see Methods). We tested all combinations of these different hyperparameters, for a total of 256 networks (Fig. 3a).

We found that the number of clusters differed widely across networks that successfully learned all 20 tasks, ranging from the lowest (2) to the highest number (30) allowed by the clustering algorithm used (Fig. 3b). Surprisingly, the most prominent factor determining the number of clusters was the neuronal activation function used (Fig. 3c). Most networks (>80%) with Softplus and ReLU activation functions gave rise to more than five clusters. In contrast, about 80% of the networks with ReTanh and Tanh activation functions resulted in the minimum number of clusters. The network architecture, initialization and L1 weight and rate regularizations did not affect the number of clusters as substantially (Fig. 3d–g). These findings show that neural networks trained for many tasks do not necessarily develop clusters of units, but tend to do so when realistic non-saturating activation functions[28,29] are used. The reason for this discrepancy remains to be elucidated.

Relationships between neural representations of pairs of tasks. In the following sections, we will focus our analyses on the reference networks. The map of normalized task variance in Fig. 2c allowed us to visualize the whole network across many tasks all at once. However, it is of limited use when we try to compare with experimental data or to analyze the (dis)similarity of the neural task representation between any pair of tasks. To quantify how each unit is selective in one task in comparison to another task, we introduce a simple measure based on task variance: the fractional task variance (FTV). For unit i, the FTV with respect to task A and task B is defined as

$$\mathrm{FTV}_i(A,B)=\frac{\mathrm{TV}_i(A)-\mathrm{TV}_i(B)}{\mathrm{TV}_i(A)+\mathrm{TV}_i(B)}$$

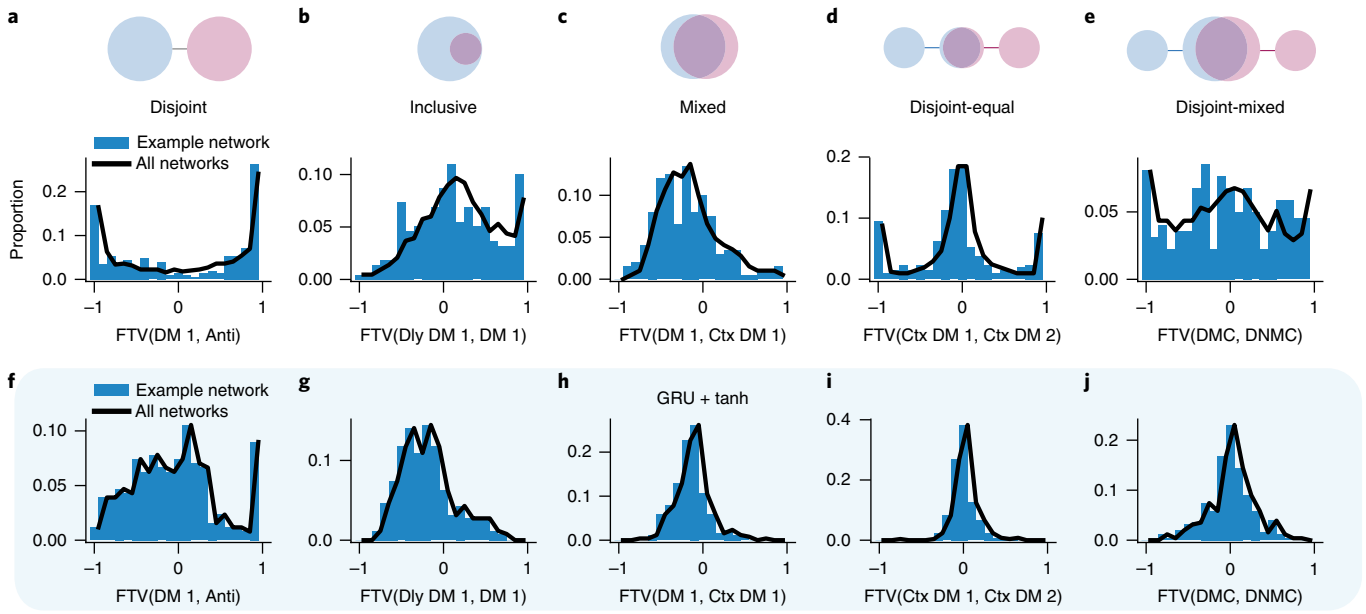

Fig. 4 | A diversity of neural relationships between pairs of tasks. For a pair of tasks, we characterized their neural relationship by the distribution of FTV over all units. **a**–**e**, In networks with the Softplus activation function, we observed five typical relationships: disjoint (**a**), inclusive (**b**), mixed (**c**), disjoint-equal (**d**), and disjoint-mixed (**e**). Blue: distribution for one example network. Black: averaged distribution over 20 networks. **f**–**j**, In networks with the Tanh activation function and the leaky GRU architecture (blue shaded background), the FTV distributions were largely mixed or equal for the same pairs of tasks. The pairs of tasks analyzed were DM1 and Anti (**a**,**f**), Dly DM 1 and DM 1 (**b**,**g**), DM 1 and Ctx DM 1 (**c**,**h**), Ctx DM 1 and Ctx DM 2 (**d**,**i**), or DMC and DNMC (**e**,**j**). Results from networks with the leaky RNN architecture and Tanh activation function are not shown because none of them learned all tasks.

where $TV_i(A)$ and $TV_i(B)$ are the task variances for tasks A and B, respectively. FTV ranges between −1 and +1. Having a $FTV_i(A,B)$ close to +1 (or −1) means that unit i is primarily selective in task A (or B).

For every pair of tasks, we were able to compute the FTV for all units that were active in at least one of the two tasks. Each distribution of FTVs contained rich information about the single-unit level neural relationship between the pair of tasks. Having 20 tasks provided us with 190 distinct FTV distributions (Supplementary Fig. 6), from the shape of which we informally summarized five typical neural relationships (Fig. 4).

1. Disjoint (Fig. 4a). When two tasks have a disjoint relationship such as the Anti task and the DM1 task, the FTV distribution was characterized by two peaks at the two ends and few units in between. There was little overlap between units selective in the two tasks. The shape of the FTV distribution was rather robust across independently trained networks: the FTV distribution from one sample network closely matches the averaged distribution from 20 networks.

2. Inclusive (Fig. 4b). This relationship was embodied by a strongly skewed FTV distribution, suggesting that one task is neurally a subset of another task. In this case, there were no units that were selective in the DM1 task but not in the Dly DM 1 task.

3. Mixed (Fig. 4c). A mixed relationship was characterized by a broad unimodal FTV distribution centered around zero with no clear peak at the two ends. This distribution suggests that the two tasks use overlapping neural circuits.

4. Disjoint-equal (Fig. 4d). For Ctx DM 1 and 2, the FTV distribution was trimodal, with two peaks at the two ends and another peak around zero. This relationship can be considered to be a combination of the disjoint relationship and the equal relationship. The equal relationship is represented by a single, narrow peak around zero. In this scenario, the two tasks each get a private neural population and share the third population.

5. Disjoint-mixed (Fig. 4e). This relationship is a combination of the disjoint and the mixed relationships. Many units only participated in one of the two tasks, whereas the rest of the units were mixed in both tasks.

Consistent with the findings above (Fig. 3), networks with the Tanh activation function showed very different FTV distributions. In such networks, the FTV distribution for a pair of tasks typically involved a single narrow peak, indicating that units were involved with similar strengths in both tasks (Fig. 4f–j).

In summary, we introduced a simple yet informative measure to study the potentially diverse neural relationships between pairs of tasks. We found that, in the reference networks, these relationships could be categorized into several typical classes. FTV distributions could be easily computed using neural data, facilitating comparisons with experiments on pairwise neural relationships between tasks.

Dissecting the circuit for the context-dependent DM tasks. Here we 'open the black box'[31] and demonstrate how an example network could be dissected and analyzed based on its clusters in two sample cognitive tasks. Context-dependent DM tasks[11] require selective processing, integration and memory of sensory inputs. Our set of tasks includes two such tasks, Ctx DM 1 and Ctx DM 2 (Supplementary Table 1). Three subgroups of units emerged in the network (Fig. 5a). Group 1 (2) units were primarily engaged in context 1 (2), whereas group 12 (one-two) units were engaged equally in both contexts. Inactivating or 'lesioning' all group 1 (2) units at once resulted in a failure in performing the Ctx DM 1 (2) tasks, respectively (Fig. 5b and Supplementary Fig. 7). In contrast, lesioning group 12 units impaired performance across all DM tasks. These results suggest that, in this network, groups 1 and 2 are responsible for selective processing of sensory inputs, whereas group 12 is critical for DM.

We next studied the connection weights of groups 1, 2, and 12 units to understand their roles. Sensory inputs from modality 1 sent strongly tuned projections to group 1 units, but not to group 2 units (Fig. 5c). Group 12 units also received direct, tuned sensory

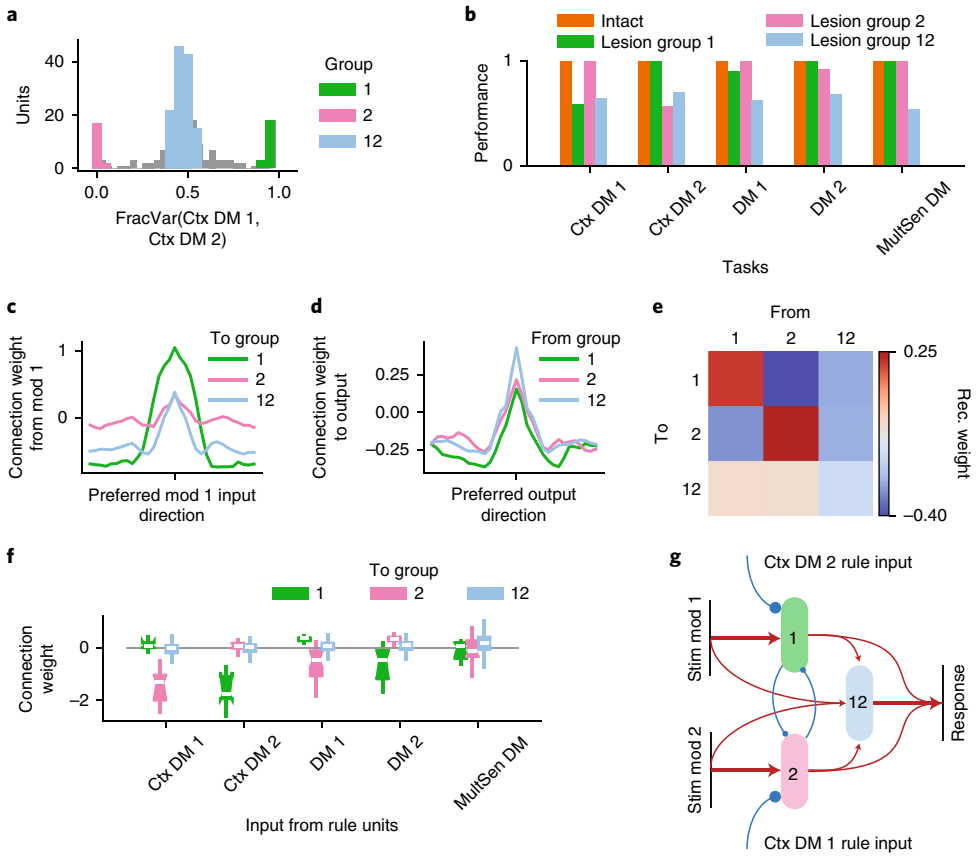

Fig. 5 | Dissecting a reference network for the context-dependent DM tasks. a, The FTV distribution for the Ctx DM 1 and 2 tasks in an example network. Most units are segregated into three groups on the basis of their FTV values. **b**, After lesioning all group 1 units together (green), the network could no longer perform the Ctx DM 1 task, whereas performance for other tasks remained intact. Instead, lesioning all group 12 units disrupted the performance for all DM tasks. **c**,**d**, Average connections from modality 1 input units to recurrent units (**c**) and from recurrent units to output units (**d**). Modality 1 input units made strongly tuned projections to group 1 units. Input and output connections are sorted by each unit's preferred input and output direction, respectively, defined as the direction represented by the strongest weight. **e**, Network wiring architecture that emerged from training, in which group 1 and group 2 units excited themselves and strongly inhibited each other. Both group 1 and 2 units excited group 12 units. Rec, recurrent. **f**, Group 1 (2) units received strong negative connections from rule units representing the Ctx DM 2 (1) task. The boxplot shows the median (horizontal line), the confidence interval of the median obtained with bootstrapping (notches), lower and upper quartile values (box), and the range of values (whisker). **g**, Cluster-based circuit diagram summarizing the neural mechanism of the Ctx DM tasks in the reference network.

inputs from both modalities. All groups projected to the output units (Fig. 5d), although group 12 units were more critical for DM (Fig. 5b). Group 1 and group 2 units excited themselves while inhibiting each other. Both projected to group 12 units (Fig. 5e). Group 1 units received negative connection weights from the rule input units representing the Ctx DM 2 task (Fig. 5f), which explained why group 1 units are silent during context 2. In summary, from training emerged two groups of units that were specialized for selective input processing. Along with the sensory inputs from both modalities, both groups fed into the third group that was specialized for DM (Fig. 5g). Our dissection of a reference network here relies on the existence of clusters. Analyzing networks with no cluster, state-space, and dynamical-system approaches may be more appropriate[11].

Compositional representations of tasks. A cognitive task can, in general, be expressed abstractly as a sequence of sensory, cognitive and motor processes, and cognitive processes may involve a combination of basic functions (such as working memory) required to perform the task (Supplementary Table 1). The compositionality of cognitive tasks is natural for human subjects because tasks are instructed with natural languages, which are compositional in nature[12]. For example, the Go task can be instructed as 'saccade to the direction of the stimulus after the fixation cue goes off', whereas the Dly Go task can be instructed as 'remember the direction of the stimulus, then saccade to that direction after the fixation cue goes off'. Therefore, the Dly Go task can be expressed as a composition of the Go task with a particular working memory process. Similarly, the Anti task can be combined with the same working memory process to form the Dly Anti task.

Here, we tested whether the reference network developed a simple algebraic form of compositional representations for tasks, even when it was never explicitly provided with the relationships between tasks. We studied the representation of each task as a single high-dimensional vector. To compute this 'task vector', we averaged neural activities across all possible stimulus conditions in each task and focused on the steady-state response during the stimulus epoch (Fig. 6a). Thus, the neural population state near the end of stimulus presentation was able to represent how the network processed the stimulus in a particular task to meet the computational need of subsequent behavioral epochs. Indeed, this idea was confirmed using principal component analysis, which revealed that task vectors in the state space spanned by the top two principal components were distinct for all 20 tasks (Supplementary Fig. 8).

We found that the vector pointing from the Go task vector toward the Dly Go task vector was very similar to the vector pointing from the Anti vector to the Dly Anti vector (Fig. 6b). This finding was

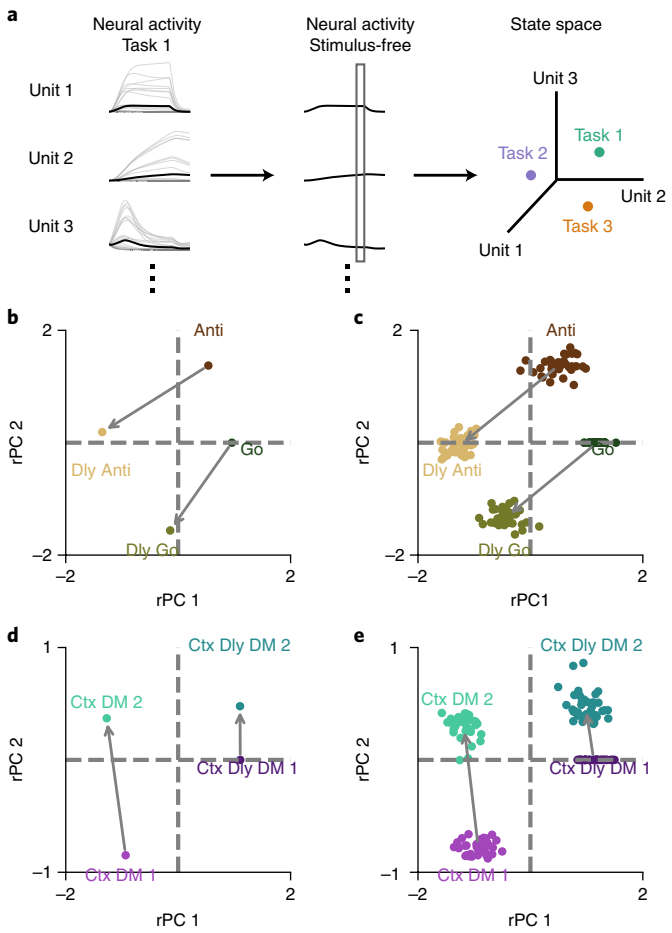

Fig. 6 | Compositional representation of tasks in state space. a, The representation of each task is the population activity of the recurrent network at the end of the stimulus presentation, averaged across different stimulus conditions (black). Gray curves indicate the neural activities in individual task conditions. **b**, Representations of the Go, Dly Go, Anti, and Dly Anti tasks in the space spanned by the top two principal components (PCs) for a sample network. For better comparison across networks, the top two PCs are rotated and reflected (rPCs) to form the two axes (see Methods). **c**, The analysis described in **b** was performed for 20 networks, and the results are overlaid. **d**, Representations of the Ctx DM 1, Ctx DM 2, Ctx Dly DM 1, and Ctx Dly DM 2 tasks in the top two PC for a sample network. **e**, The analysis described in **d** was performed for $n = 40$ independent networks.

robust across many independently trained networks (Fig. 6c). The Go-to-Dly Go vector and the Anti-to-Dly Anti vector presumably reflect the cognitive process of working memory. Similar findings were made with another set of tasks. The vector pointing from the Ctx DM 1 task to the Ctx DM 2 task was similar to the vector pointing from the Ctx Dly DM 1 task to the Ctx Dly DM 2 task (Fig. 6d,e). The Ctx DM 1 to Ctx DM 2 vector reflects the difference between the gating modality 1 and the gating modality 2 processes. These results suggest that the representation of a task can potentially be expressed as the algebraic sum of vectors representing the underlying sensory, cognitive, and motor processes. This finding is reminiscent of previous work showing that neural networks can represent words and phrases compositionally[32].

Performing tasks with composition of rule inputs. We showed that the representation of tasks could be compositional in principle. However, it is unclear whether in our reference network this principle of compositionality can be extended from representing to performing tasks. The network is normally instructed which task to perform by activation of the corresponding rule input unit. What would the network do in response to a compositional rule signal as a combination of several activated and deactivated rule units? We tested whether the network can perform tasks by receiving composite rule inputs (Fig. 7a).

Consider the same two sets of tasks as in Fig. 6. The network was able to perform the Dly Anti task well when provided with the particular combination of rule inputs: Anti + (Dly Go − Go) (Fig. 7b). In contrast, the network failed to perform the Dly Anti task when provided with several other combinations of rule inputs (Fig. 7b). Similarly, the network can perform the Ctx Dly DM 1 task best when provided the composite rule inputs of Ctx Dly DM 2 + (Ctx DM 1 − Ctx DM 2) (Fig. 7c). In accordance with these results, we found that connection weights from individual rule input units to recurrent units also displayed an algebraic compositional structure (Supplementary Fig. 9). Similar results were found when each rule activates and inactivates a distributed set of rule units (Supplementary Fig. 10). However, similar success was not obtained for the family of matching tasks (DMS, DNMS, DMC, DNMC) (Supplementary Table 1). The network cannot perform the DMS task when provided with the composite rule inputs DMC + DNMS − DNMC (Supplementary Fig. 11). These results indicated that the reference networks learned several, but not all, implicit compositional relationships between tasks.

In a network with this simple form of compositionality, acquiring a new task may need no modification to the recurrent connections because all components for this task have already been learned. Instead, a new task may be acquired by learning the appropriate rule input that controls the information flow in the network[2]. To test these hypotheses, we studied how well networks can learn new tasks when pre-trained on a set of tasks. Networks were able to learn a new task substantially faster when pre-trained on tasks with similar components (Fig. 7d). A network pre-trained on related tasks was even able to learn a new task by modifying only the connection weights from rule input units to the recurrent network (Fig. 7e). Preliminary analysis showed that Ctx DM-pre-trained networks slowly learned the Dly Anti task (Fig. 7e) by engaging recurrent units that have different preferences for stimulus and response directions.

Continual task training alters the neural representation. In most of the networks presented so far, all tasks were randomly interleaved during training, and the networks adjusted all of the connection weights to perform the 20 tasks optimally. However, adult animals typically learn tasks sequentially. When an adult animal is learning some new tasks, its brain needs to implicitly balance the need of learning with the need of retaining past memories. Otherwise, the brain could suffer from a common problem in artificial neural networks known as 'catastrophic forgetting'. Learning new tasks will happen at the expense of forgetting previous memories. The network learns to perform a task by modifying parameters to minimize the loss function for this task. During sequential training of two tasks, if the network uses traditional learning techniques, training for the second task can easily result in the failure of performing the first task, as the minima of the loss functions of tasks 1 and 2 are far apart (Fig. 8a, gray line).

Continual-learning techniques can protect previously learned tasks by preventing large changes of important network parameters (Fig. 8a, red line). Here we distinguish sequential training, which describes the task presentation, from continual learning, which refers to particular learning algorithms. Several continual-learning methods have recently been proposed to battle catastrophic forgetting[33–35]. These methods typically involve selective protection of

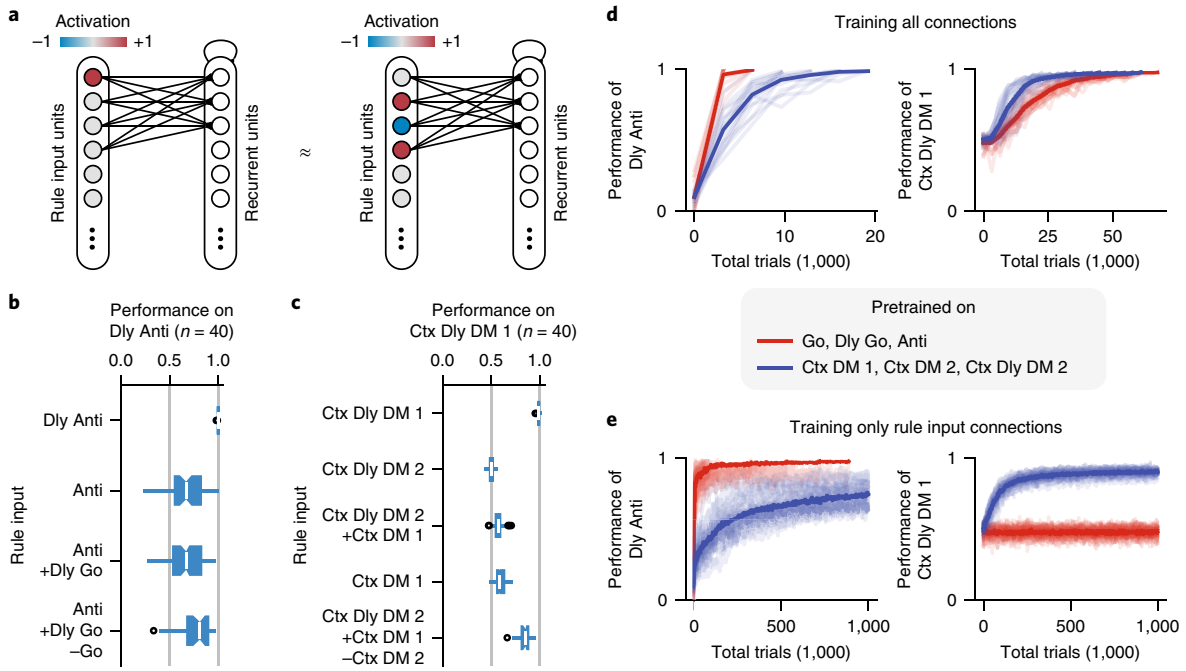

Fig. 7 | Performing tasks with algebraically composite rule inputs. a, During training, a task is always instructed by activation of the corresponding rule input unit (left). After training, the network can potentially perform a task by activation or deactivation of a set of rule input units meant for other tasks (right). **b**, The network can perform the Dly Anti task well if given the Dly Anti rule input or the Anti + (Dly Go − Go) rule input. The network fails to perform the Dly Anti task when provided other combinations of rule inputs. **c**, Similarly, the network can perform the Ctx Dly DM 1 task well when provided with the Ctx Dly DM 2 + (Ctx DM 1 − Ctx DM 2) rule input. Circles represent the results of individual networks and bars represent median performances of 40 networks. The boxplot convention in **b**,**c** is the same as the one in Fig. 5f. **d**, Left, network performance during training of the Dly Anti task when the network is pre-trained on Go, Dly Go, and Anti tasks (red), or the Ctx DM 1, Ctx DM 2, and Ctx Dly DM 2 tasks (blue). Right, network performance during training of the Ctx Dly DM 1 task under the same pre-training conditions. Individual networks (light); mean across 40 networks (bold). All connections are adjusted during training. **e**, Similar to **d**, but only training the rule input connections in the second training phase.

connection weights that are deemed important for previously learned tasks. By employing one such technique[35], we were able to substantially improve the performance of networks that were sequentially trained on a set of cognitive tasks (Fig. 8b,c). Notably, all of the networks were initialized with random connection weights. The tasks were made easier to facilitate training (see Methods). The continual-learning technique was especially effective at helping the network retain performance of tasks learned earlier. For example, the continual-learning networks can retain high performance in a working memory task (Dly Go) after successfully learning five extra tasks (DM1, DM2, MultSen DM, Ctx DM 1, and Ctx DM 2) (Fig. 8b). An example network achieved high performance on the DM1, DM2, and MultSen DM tasks even before being trained on them, suggesting that these tasks rely on structures that can be learned through other tasks. In addition, continual learning resulted in a much slower acquisition of difficult tasks (Ctx DM 1 and 2) (Fig. 8c).

To understand the impact of continual learning on neural representation, we analyzed the FTV distributions for the Ctx DM 1 and 2 tasks in the sequentially trained networks. The shapes of the FTV distributions were markedly different across networks trained with or without the continual-learning technique (Fig. 8d). Instead of a strongly trimodal distribution, the FTV distribution of a continual-learning network contained a strong peak at the middle, with only minor peaks at the two ends.

The continual-learning technique works by selective protection of connection weights, and we can directly mimic this effect by training networks that are partially plastic[17]. In contrast with the fully plastic networks, the FTV distributions for the Ctx DM 1 and 2 tasks in the partially plastic networks were again mainly unimodal (Supplementary Fig. 12a).

Finally, we analyzed single- and multi-unit recordings from the frontal eye field of macaque monkeys performing similar context-dependent DM tasks[11] (Fig. 8e and Supplementary Fig. 12b–e). The FTV distribution derived from experimental data mainly consists of a broad, unimodal distribution, consistent with networks sequentially trained using a continual-learning technique (Fig. 8d, red) and with partially plastic networks (Supplementary Fig. 12a, dark blue). These findings suggest that adult brains do not necessarily develop the 'optimal' circuit-level solution for newly learned tasks, even if they have been trained for months on the same tasks. Instead, the brain may balance the need between performing the tasks at hand and retaining past memories.

Discussion

Higher-order cortical areas, especially the lateral prefrontal cortex, are markedly versatile in their engagement in a wide range of cognitive functions. Here we investigated how multiple cognitive tasks are represented in various RNN models. In the networks with non-saturating activities, we identified clusters of units that were each specialized for a subset of tasks. Each cluster potentially represents a particular sequence of the sensori-motor events and a subset of cognitive processes that serve as these networks' building blocks for flexible behavior. We showed that realistic neuronal activation functions[28,29] robustly led to clustering across a wide range of networks. In addition, we found that the representation of tasks in our network showed a form of compositionality, a critical feature for cognitive flexibility. By virtue of the compositionality, a task can be correctly instructed by composing instructions for other tasks. Finally, using a recently proposed continual-learning technique, we were able to train networks to learn many tasks sequentially.

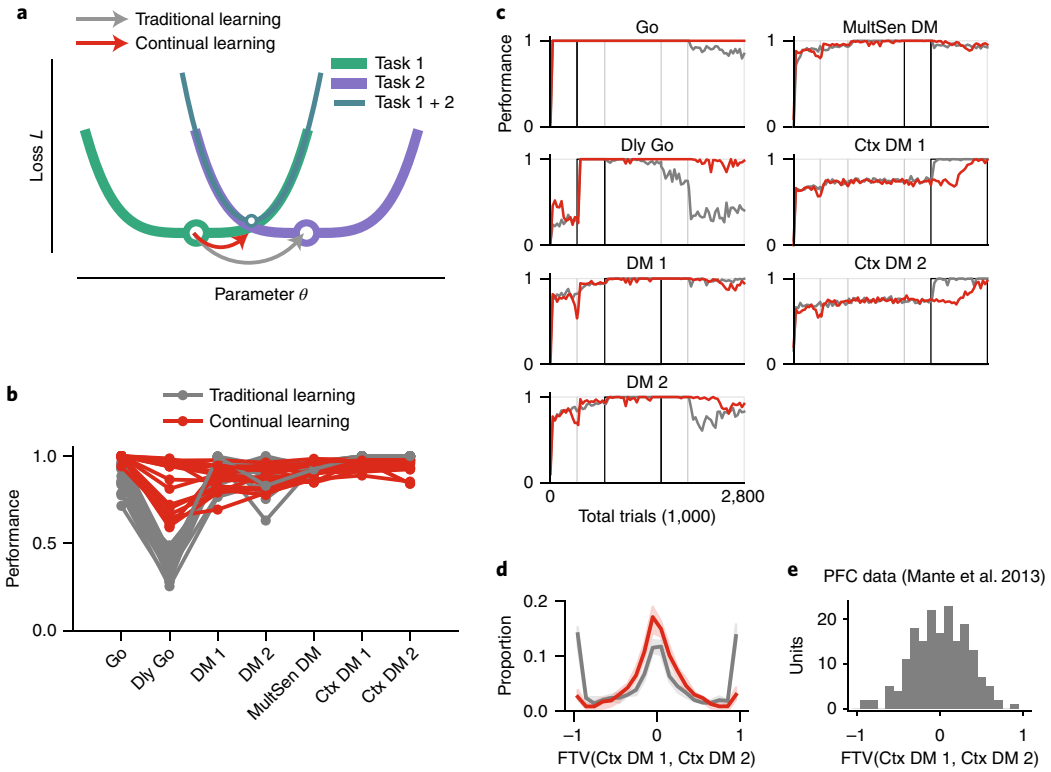

Fig. 8 | Sequential training of cognitive tasks. a, Schematics of continual learning compared with traditional learning. Network parameters (such as connection weights) optimal for a new task can be destructive for old tasks. Arrows show changes of an example parameter θ when task 2 is trained after task 1 is already learned. **b**, Final performance across all trained tasks with traditional (gray) or continual (red) learning techniques. Lines represent the results of individual networks. Only networks that achieved more than 80% accuracy on Ctx DM 1 and 2 are shown. **c**, Performance of all tasks during sequential training. Example networks used traditional (gray) or continual (red) learning techniques, respectively. For each task, the black box indicates the period in which this task was trained. DM 1 and 2 tasks were trained in the same block to prevent bias, as were Ctx DM 1 and 2 tasks. **d**, FTV distributions for networks with traditional (gray) or continual (red) learning techniques. Solid lines are median over 20 networks. Shaded areas indicate the 95% confidence interval of the median estimated from bootstrapping. **e**, The FTV computed using single-units data from the prefrontal cortex of a monkey performing Ctx DM 1 and 2 (ref. 11).

The FTV distributions in continual-learning networks were substantially more mixed, and are consistent with those computed from prefrontal data in monkeys performing similar tasks.

Functional specialization is one of the most fundamental design principle of the brain[36], dating back to Broca's area. The brain is partitioned into specialized areas, and each brain area consists of genetically and functionally distinct cell types. In contrast, theoretical studies argued that for maximum cognitive flexibility, prefrontal neurons should be selective to mixtures of multiple task variables[37]. Mixed selectivity neurons are indeed ubiquitous inside the prefrontal cortex[10,23]. These findings pose a challenging conceptual question: when is functional specialization desired, and when is it not? Our results suggest a tentative unifying answer to this question. For a computation that is repeatedly performed and shared in various behaviors, it is computationally beneficial to develop a functionally specialized circuit for it. In neural networks, functional specialization can be achieved by training multiple inter-related tasks simultaneously. In biological brains, specialization is potentially achieved through evolution and development, instead of learning in adulthood. Mixed selectivity, on the other hand, can be beneficial to a neural system that needs to maintain both flexibility for future learning and memories from the past. The process of learning new cognitive tasks in adult brains is probably better modeled in neural networks using a continual-learning paradigm.

The neural mechanism behind multiple cognitive tasks has been investigated in human imaging studies. In a series of experiments, Cole and colleagues trained humans to perform 64 cognitive tasks following compositional rule instructions[6,38]. They trained linear classifiers to decode rules from prefrontal neural activity patterns. These classifiers can substantially generalize to novel tasks[6], consistent with a compositional neural representation of rules. Although trained with discrete rule instructions, our reference network develops a clear compositional structure in its representations for two sets of tasks, as shown using the population activity. However, it is unlikely that our network possesses a more general form of compositionality, which requires that task components can be arbitrarily and recursively combined to perform complex new tasks. Indeed, our network is not able to perform the DMS task using composite rule inputs; more broadly, it remains unclear whether standard modern recurrent network architectures can accomplish challenging compositional tasks[39,40].

Similar to other works on trained neural networks[11,14–19,41], the machine learning protocol we used is not validated biologically. Furthermore, in our network, a rule input is explicitly provided throughout the trial, therefore there is no need for the network to hold the 'task set' internally using persistent activity[4,5]. This, however, can be remedied by providing the rule cue only at the beginning of each trial, which would encourage the network to internally sustain the task set. We can even ask the network to figure out a task rule by trial-and-error[42]. In spite of these concerns, our approach offers an efficient computational platform to test hypotheses about neural representations and mechanisms that could guide experiments and data analysis. Future progress in this direction, at the interface between neuroscience and artificial intelligence, will advance our understanding of flexible behavior in many cognitive tasks.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, statements of data availability and associated accession codes are available at https://doi.org/10.1038/s41593-018-0310-2.

Received: 16 July 2018; Accepted: 30 November 2018;
Published online: 14 January 2019

References

1. Fuster, J. *The Prefrontal Cortex* (Academic Press, Cambridge, 2015).
2. Miller, E. K. & Cohen, J. D. An integrative theory of prefrontal cortex function. *Annu. Rev. Neurosci.* **24**, 167–202 (2001).
3. Wang, X.-J. in *Principles of Frontal Lobe Function* (Stuss, D. T. & Knight, R. T. eds.) (Cambridge Univ. Press, New York, 2013).
4. Wallis, J. D., Anderson, K. C. & Miller, E. K. Single neurons in prefrontal cortex encode abstract rules. *Nature* **411**, 953–956 (2001).
5. Sakai, K. Task set and prefrontal cortex. *Annu. Rev. Neurosci.* **31**, 219–245 (2008).
6. Cole, M. W., Etzel, J. A., Zacks, J. M., Schneider, W. & Braver, T. S. Rapid transfer of abstract rules to novel contexts in human lateral prefrontal cortex. *Front. Hum. Neurosci.* **5**, 142 (2011).
7. Tschentscher, N., Mitchell, D. & Duncan, J. Fluid intelligence predicts novel rule implementation in a distributed frontoparietal control network. *J. Neurosci.* **37**, 4841–4847 (2017).
8. Hanes, D. P., Patterson, W. F. II & Schall, J. D. Role of frontal eye fields in countermanding saccades: visual, movement, and fixation activity. *J. Neurophysiol.* **79**, 817–834 (1998).
9. Padoa-Schioppa, C. & Assad, J. A. Neurons in the orbitofrontal cortex encode economic value. *Nature* **441**, 223–226 (2006).
10. Rigotti, M. et al. The importance of mixed selectivity in complex cognitive tasks. *Nature* **497**, 585–590 (2013).
11. Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013).
12. Cole, M. W., Laurent, P. & Stocco, A. Rapid instructed task learning: a new window into the human brain's unique capacity for flexible cognitive control. *Cogn. Affect. Behav. Neurosci.* **13**, 1–22 (2013).
13. Reverberi, C., Görgen, K. & Haynes, J.-D. Compositionality of rule representations in human prefrontal cortex. *Cereb. Cortex* **22**, 1237–1246 (2012).
14. Zipser, D. & Andersen, R. A. A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature* **331**, 679–684 (1988).
15. Song, H. F., Yang, G. R. & Wang, X.-J. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS Comput. Biol.* **12**, e1004792 (2016).
16. Carnevale, F., de Lafuente, V., Romo, R., Barak, O. & Parga, N. Dynamic control of response criterion in premotor cortex during perceptual detection under temporal uncertainty. *Neuron* **86**, 1067–1077 (2015).
17. Rajan, K., Harvey, C. D. & Tank, D. W. Recurrent network models of sequence generation and memory. *Neuron* **90**, 128–142 (2016).
18. Chaisangmongkon, W., Swaminathan, S. K., Freedman, D. J. & Wang, X.-J. Computing by robust transience: how the fronto-parietal network performs sequential, category-based decisions. *Neuron* **93**, 1504–1517 (2017).
19. Eliasmith, C. et al. A large-scale model of the functioning brain. *Science* **338**, 1202–1205 (2012).
20. Funahashi, S., Bruce, C. J. & Goldman-Rakic, P. S. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *J. Neurophysiol.* **61**, 331–349 (1989).
21. Gold, J. I. & Shadlen, M. N. The neural basis of decision making. *Annu. Rev. Neurosci.* **30**, 535–574 (2007).
22. Siegel, M., Buschman, T. J. & Miller, E. K. Cortical information flow during flexible sensorimotor decisions. *Science* **348**, 1352–1355 (2015).
23. Raposo, D., Kaufman, M. T. & Churchland, A. K. A category-free neural population supports evolving demands during decision-making. *Nat. Neurosci.* **17**, 1784–1792 (2014).
24. Romo, R., Brody, C. D., Hernández, A. & Lemus, L. Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature* **399**, 470–473 (1999).
25. Munoz, D. P. & Everling, S. Look away: the anti-saccade task and the voluntary control of eye movement. *Nat. Rev. Neurosci.* **5**, 218–228 (2004).
26. Miller, E. K., Erickson, C. A. & Desimone, R. Neural mechanisms of visual working memory in prefrontal cortex of the macaque. *J. Neurosci.* **16**, 5154–5167 (1996).
27. Freedman, D. J. & Assad, J. A. Neuronal mechanisms of visual categorization: an abstract view on decision making. *Annu. Rev. Neurosci.* **39**, 129–147 (2016).
28. Priebe, N. J. & Ferster, D. Inhibition, spike threshold, and stimulus selectivity in primary visual cortex. *Neuron* **57**, 482–497 (2008).
29. Abbott, L. F. & Chance, F. S. Drivers and modulators from push-pull and balanced synaptic input. *Prog. Brain. Res.* **149**, 147–155 (2005).
30. Wang, X.-J. Probabilistic decision making by slow reverberation in cortical circuits. *Neuron* **36**, 955–968 (2002).
31. Sussillo, D. & Barak, O. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Comput.* **25**, 626–649 (2013).
32. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural. Inf. Process. Syst.* **26**, 3111–3119 (2013).
33. Benna, M. K. & Fusi, S. Computational principles of synaptic memory consolidation. *Nat. Neurosci.* **19**, 1697–1706 (2016).
34. Kirkpatrick, J. et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl Acad. Sci. USA* **114**, 3521–3526 (2017).
35. Zenke, F., Poole, B. & Ganguli, S. Continual learning through synaptic intelligence. *ICML* **70**, 3987–3995 (2017).
36. Kanwisher, N. Functional specificity in the human brain: a window into the functional architecture of the mind. *Proc. Natl Acad. Sci. USA* **107**, 11163–11170 (2010).
37. 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. *Front. Comput. Neurosci.* **4**, 24 (2010).
38. Cole, M. W. et al. Multi-task connectivity reveals flexible hubs for adaptive task control. *Nat. Neurosci.* **16**, 1348–1355 (2013).
39. Yang, G. R., Ganichev, I., Wang, X.-J., Shlens, J. & Sussillo, D. A dataset and architecture for visual reasoning with a working memory. ECCV 714–731 (2018)..
40. Lake, B. M. & Baroni, M. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *ICML* **80**, 2873–2882 (2017).
41. Yamins, D. L. K. et al. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proc. Natl Acad. Sci. USA* **111**, 8619–8624 (2014).
42. Song, H. F., Yang, G. R. & Wang, X.-J. Reward-based training of recurrent neural networks for cognitive and value-based tasks. *eLife* **6**, e21492 (2017).

Acknowledgements

We thank current and former members of the Wang lab, especially S.Y. Li, O. Marschall, and E. Ohran for fruitful discussions; J.A. Li, J.D. Murray, D. Ehrlich, and J. Jaramillo for critical comments on the manuscript; and S. Wang for assistance with the NYU HPC clusters. We are grateful to V. Mante for providing data and for discussion. This work was supported by an Office of Naval Research grant no. N00014-13-1-0297, a National Science Foundation grant no. 16-31586, a Google Computational Neuroscience Grant (X.J.W.), a Samuel J. and Joan B. Williamson Fellowship, a National Science Foundation Grant Number 1707398, and the Gatsby Charitable Foundation (G.R.Y.).

Author contributions

G.R.Y. and X.J.W. designed the study. G.R.Y., M.R.J., H.F.S, W.T.N., and X.J.W. had frequent discussions. G.R.Y. and M.R.J. performed the research. G.R.Y., H.F.S, W.T.N., and X.J.W. wrote the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41593-018-0310-2.

Reprints and permissions information is available at www.nature.com/reprints.

Correspondence and requests for materials should be addressed to X.-J.W.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Methods

Network structure. The RNNs shown in the main text all contain $N_{rec} = 256$ units. The results are largely insensitive to the network size. Similar results were obtained in networks of sizes between 128 and 512 units (the range we tested). Every network is a time-discretized RNN with positive activity[15]. Before time discretization, the network activity $\mathbf{r}$ follows a continuous dynamical equation

$$\tau \frac{d\mathbf{r}}{dt} = -\mathbf{r} + f(W^{rec}\mathbf{r} + W^{in}\mathbf{u} + \mathbf{b} + \sqrt{2\tau\sigma_{rec}^2}\boldsymbol{\xi})$$

In this equation, $\tau = 100$ ms is the neuronal time constant. Real neurons typically have shorter time constants around 20 ms, here the 100 ms time constant mimics the slower synaptic dynamics on the basis of NMDA receptors[30]. $\mathbf{u}$ is the input to the network, $\mathbf{b}$ is the bias or background input, $f(\cdot)$ is the neuronal nonlinearity, $\boldsymbol{\xi}$ are N_{rec} independent Gaussian white noise processes with zero mean and unit variance and $\sigma_{rec} = 0.05$ is the strength of the noise. In the reference setting, we use a standard Softplus function

$$f(x) = \log(1 + \exp(x))$$

which after re-parameterization is very similar to a neuronal nonlinearity, that is, the frequency-current curve, commonly used in previous neural circuit models[29]. A set of output units $\mathbf{z}$ read out nonlinearly from the network,

$$\mathbf{z} = g(W^{out}\mathbf{r})$$

where $g(x) = 1/(1 + \exp(-x))$ is the logistic function, bounding output activities between 0 and 1. W^{in}, W^{rec}, W^{out} are the input, recurrent and output connection matrices, respectively.

After using the first-order Euler approximation with a time-discretization step Δt, we have

$$\mathbf{r}_t = (1-\alpha)\mathbf{r}_{t-1} + \alpha f(W^{rec}\mathbf{r}_{t-1} + W^{in}\mathbf{u}_t + \mathbf{b} + \sqrt{2\alpha^{-1}\sigma_{rec}^2}\mathbf{N}(0,1))$$

Here $\alpha \equiv \Delta t/\tau$, and N(0,1) stands for the standard normal distribution. We use a discretization step $\Delta t = 20$ ms. We imposed no constraint on the sign or the structure of the weight matrices W^{in}, W^{rec}, W^{out}. The network and the training are implemented in TensorFlow.

The network receives four types of noisy input,

$$\boldsymbol{u} = (u_{fix}, \boldsymbol{u}_{mod1}, \boldsymbol{u}_{mod2}, \boldsymbol{u}_{rule}) + \boldsymbol{u}_{noise}$$
$$\boldsymbol{u}_{noise} = \sqrt{2/\alpha}\sigma_{in}\mathbf{N}(0,1)$$

Here the input noise strength $\sigma_{in} = 0.01$. The fixation input u_{fix} is typically at the high value of 1 when the network should fixate. The fixation input goes to 0 when the network is required to respond. The stimulus inputs $\mathbf{u}_{mod1}$ and $\mathbf{u}_{mod2}$ comprise two 'rings' of units, each representing a one-dimensional circular variable described by the degree around a circle. Each ring contains 32 units, whose preferred directions are uniformly spaced from 0 to 2π. For unit i with a preferred direction θ_i, its activity for a stimulus presented at direction ψ is

$$u_i = \gamma \cdot 0.8 \exp\left[-\frac{1}{2}\left(\frac{8\ |\psi - \psi_i|}{\pi}\right)^2\right]$$

where γ is the strength of the stimulus. For multiple stimuli, input activities are added together. The network also receives a set of rule inputs $\mathbf{u}_{rule}$ that encode which task the network is supposed to perform on each trial. Normally, $\mathbf{u}_{rule}$ is a one-hot vector. That means the rule input unit corresponding to the current task is activated at 1, while other rule input units remain at 0. Therefore, the number of rule input units equals to the number of tasks trained. For compositional rule inputs (Fig. 7), the activation of rule input units can be an arbitrary pattern. For example, for the combined rule input Anti + (Dly Go − Go), the activities of the rule input units corresponding to the Go, Dly Go and Anti tasks are −1, +1 and +1, respectively. For Supplementary Fig. 10, each rule activates/inactivates a distributed set of 20 rule input units. The rule unit activation patterns for different rules are orthogonal to each other. They are chosen from rows of a random orthogonal matrix, generated using the Python package scipy.stats.ortho_group. In total, there are $N_{in} = 1 + 32 \times 2 + 20 = 85$ input units.

The network projects to an output ring $\mathbf{z}_{out}$, which also contains 32 units. The output ring units encode the response directions using similar tuning curves to the ones used for the input rings. In addition, the network projects to a fixation output unit z_{fix}, which should be at the high activity value of 1 before the response and at 0 once a response is generated. In total, there are $N_{out} = 1 + 32 = 33$ output units.

We lesion a network unit by setting to 0 its projection weights to all recurrent and output units.

Tasks and performances. Here we first describe the common setup for the 20 tasks trained. Deviations from the common setup will be described below individually.

The rule input unit corresponding to the current task will be activated throughout the whole trial. The network receives a fixation input, which is activated from the beginning of the trial. When the fixation input is on, the network should fixate by having the fixation output unit at a high activity $\hat{z}_{fix} = 0.85$. The offset of the fixation input usually indicates the onset of the response or go epoch, when the network needs to report the response direction through activities of the output ring. During the response epoch, the fixation output unit has a target output of $\hat{z}_{fix} = 0.05$. For a target response direction ψ, the target output activity of an output unit i is

$$\hat{z}_i = 0.8 \exp\left[-\frac{1}{2}\left(\frac{8\ |\psi - \psi_i|}{\pi}\right)^2\right] + 0.05$$

where ψ_i is the preferred response direction of unit i. When no response is required, the target output activity is fixed at $\hat{z}_i = 0.05$. The network also receives one or two stimuli. Each stimulus contains information from modality 1, 2 or both. When there is only one stimulus, the direction of the stimulus is drawn from a uniform distribution between 0 and 360°.

A trial is considered correct only if the network correctly maintained fixation and responded to the correct direction. The response direction of the network is read out using a population vector method. The decoded response direction is considered correct if it is within 36° of the target direction. If the activity of the fixation output falls below 0.5, the network is considered to have broken fixation.

The discrimination thresholds a in Supplementary Fig. 2 are obtained by fitting Weibull functions to performances p as a function of coherences c at a fixed stimulus duration,

$$p = 1 - 0.5\exp(-(c/a)^b)$$

Each task can be separated into distinct epochs. Fixation (fix) epoch is the period before any stimulus is shown. It is followed by the stimulus epoch 1 (stim1). If there are two stimuli separated in time, then the period between the two stimuli is the delay epoch and the second stimulus is shown in the stimulus epoch 2 (stim2). The period when the network should respond is the go epoch. The duration of the fixation, stim1, delay1, stim2 and go epochs are T_{fix}, T_{stim1}, T_{delay1}, T_{stim2}, T_{go}, respectively. For convenience, we grouped the 20 tasks into five task families: the Go, Anti, DM, Delayed Decision Making (Dly DM), and Matching families.

Go task family. This family of tasks includes the Go, RT Go and Dly Go tasks. In all three tasks, a single stimulus is randomly shown in either modality 1 or 2, and the response should be made in the direction of the stimulus. These three tasks differ in their stimulus onset and offset times. In the Go task, the stimulus appears before the fixation cue goes off. In the RT Go task, the fixation input never goes off, and the network should respond as soon as the stimulus appears. In the Dly Go task, a stimulus appears briefly and is followed by a delay period until the fixation cue goes off. The Dly Go task is similar to the memory-guided saccade task[20].

For the Go task,

$$T_{stim1} \sim \mathrm{U}(500, 1500)$$

$\mathrm{U}(t_1,t_2)$ is a uniform distribution between t_1 and t_2. The unit for time is milliseconds and is omitted for brevity. For the RT Go task,

$$T_{stim1} \sim \mathrm{U}(500, 2500)$$

For the Dly Go tasks,

$$T_{delay1} \sim \mathrm{U}(\{200, 400, 800, 1600\})$$

Here $\mathrm{U}(\{a_1, \ldots, a_n\})$ denotes a discrete uniform distribution over the set $\{a_1, \ldots, a_n\}$.

Anti task family. This family includes the Anti, RT Anti and Dly Anti tasks. These three tasks are the same as their counterpart Go-family tasks, except that the response should be made to the opposite direction of the stimulus.

DM family. This family includes five perceptual DM tasks: the DM 1, DM 2, Ctx DM 1, Ctx DM 2 and MultSen DM tasks. In each trial, two stimuli are shown simultaneously and are presented till the end of the trial. Stimulus 1 is drawn randomly between 0 and 360°, while stimulus 2 is drawn uniformly between 90 and 270° away from stimulus 1. In DM 1, the two stimuli only appear in modality 1, while in DM 2 the two stimuli only appear in modality 2. In DM 1 and DM 2, the correct response should be made to the direction of the stronger stimulus (the stimulus with higher γ). In Ctx DM 1, Ctx DM 2 and MultSen DM tasks, each stimulus appears in both modality 1 and 2. In the Ctx DM 1 task, information from modality 2 should be ignored, and the correct response should be made to the stronger stimulus in modality 1. In the Ctx DM 2 task, information from modality 1 should be ignored. In the MultSen DM task, the correct response should be made to the stimulus that has a stronger combined strength in modalities 1 and 2.

The DM 1 and DM 2 tasks are inspired from classical perceptual DM tasks based on random-dot motion stimuli[21]. In random-dot motion tasks, there is only one stimulus, the coherence of which is varied across trials. Following the tradition of Wang[30], we use two input stimuli to model momentary motion evidence toward the two target directions. When the two stimuli have the same strengths ($\gamma_1 = \gamma_2$), there is no net evidence toward any target direction, mimicking the condition of 0 motion coherence in the random-dot motion task. A stronger difference in the stimulus strengths emulates a stronger motion coherence. For a coherence c representing net evidence for the direction of stimulus 1, the strengths of stimulus 1 and 2 (γ_1,γ_2) are set as

$$\gamma_{1,\mathrm{mod}\ i} = \overline{\gamma} + c, \quad \gamma_{2,\mathrm{mod}\ i} = \overline{\gamma} - c$$

respectively, where $i \in \{1,2\}$ is the modality. Here, $\overline{\gamma}$ is the average strength of the two stimuli. For each trial, we draw $\overline{\gamma}$ from a uniform distribution around 1, $\overline{\gamma} \sim \mathrm{U}(0.8, 1.2)$. Indeed, in all DM-family tasks and Dly DM-family tasks, there is a single coherence c in each trial that determines the overall strength of net evidence toward the direction represented by stimulus 1. For all DM-family tasks,

$$c \sim \mathrm{U}(\{-0.08, -0.04, -0.02, -0.01, 0.01, 0.02, 0.04, 0.08\})$$

The duration of stimulus 1, which is fixed in each trial, is drawn from the following distribution,

$$T_{\mathrm{stim1}} \sim \mathrm{U}(\{400, 800, 1600\})$$

Indeed, all tasks from the DM family use the same distribution for T_{stim1}. And since the two stimuli are shown simultaneously, $T_{\mathrm{stim1}} = T_{\mathrm{stim2}}$.

The Ctx DM 1 and Ctx DM 2 tasks are inspired from context-dependent DM tasks performed by macaque monkeys[11]. Now each stimulus is presented in both modalities at the same direction, with strengths $\gamma_{1,\mathrm{mod1}}$,$\gamma_{1,\mathrm{mod2}}$ for stimulus 1, and $\gamma_{2,\mathrm{mod1}}$,$\gamma_{2,\mathrm{mod2}}$ for stimulus 2. The stimulus strengths are determined by the coherence for modality 1 and 2 (c_{mod1},c_{mod2}), so we have

$$\gamma_{1,\mathrm{mod1}} = \overline{\gamma}_{\mathrm{mod1}} + c_{\mathrm{mod1}}, \quad \gamma_{2,\mathrm{mod1}} = \overline{\gamma}_{\mathrm{mod1}} - c_{\mathrm{mod1}}$$

A similar equation holds for modality 2 as well. c_{mod1} and c_{mod2} are drawn independently from the same distribution. In Ctx DM 1, $c = c_{\mathrm{mod1}}$, while in Ctx DM 2, $c = c_{\mathrm{mod2}}$. $\overline{\gamma}_{\mathrm{mod1}}$ and $\overline{\gamma}_{\mathrm{mod2}}$ are also drawn from U(0.8,1.2). In the original Mante task[11], there is another delay period between the stimuli and the response period, which is not included here.

The MultSen DM task mimics a multi-sensory integration task[23]. The setup of stimulus is similar to those in the Ctx DM 1 and Ctx DM 2 tasks, except that the network should integrate information from both modalities and the stronger stimulus is the one with higher averaged strength from modality 1 and 2. The overall coherence $c = (c_{\mathrm{mod1}} + c_{\mathrm{mod2}})/2$. We determine all four strengths with the following procedure. First, we determine the average strength of stimulus 1 across both modalities, γ_1, and the average strength of stimulus 2, γ_2:

$$\gamma_1 = \overline{\gamma} + c, \quad \gamma_2 = \overline{\gamma} - c$$

Here, $\overline{\gamma}$ and c both follow the same distributions as other DM-family tasks. Then we set

$\gamma_{1,\mathrm{mod1}} = \gamma_1(1+\Delta_1)$, $\gamma_{1,\mathrm{mod2}} = \gamma_1(1-\Delta_1)$, where $\Delta_1 \sim \mathrm{U}(0.1, 0.4) \cup \mathrm{U}(-0.4, -0.1)$. This is similar for stimulus 2.

Dly DM family. This family includes Dly DM 1, Dly DM 2, Ctx Dly DM 1 and Ctx Dly DM 2. These tasks are similar to the corresponding tasks in the DM family, except that in the Dly DM family tasks, the two stimuli are separated in time. The Dly DM 1 and Dly DM 2 tasks are inspired by the classical parametric working memory task developed by Romo and colleagues[24]. The two stimuli are both shown briefly and are separated by a delay period. Another short delay period follows the offset of the second stimulus.

For all Dly DM family tasks,

$$T_{\mathrm{delay1}} \sim \mathrm{U}(\{200, 400, 800, 1600\})$$
$$c \sim \mathrm{U}(\{-0.32, -0.16, -0.08, 0.08, 0.16, 0.32\})$$

and $T_{\mathrm{stim1}} = T_{\mathrm{stim2}} = 300$.

Matching family. This family of tasks includes the DMS, DNMS, DMC, DNMC tasks. In these tasks, two stimuli are presented consecutively and separated by a delay period. Each stimulus can appear in either modality 1 or 2. The network response depends on whether or not the two stimuli are 'matched'. In the DMS and DNMS tasks, two stimuli are matched if they point toward the same direction, regardless of their modalities. In DMC and DNMC tasks, two stimuli are matched if their directions belong to the same category. The first category ranges from 0 to 180°, while the rest from 180 to 360° belong to the second category. In the DMS and DMC tasks, the network should respond toward the direction of the second stimulus if the two stimuli are matched and maintain fixation otherwise. In the DNMS and DNMC tasks, the network should respond only if the two stimuli are not matched, that is, a non-match, and fixate when it is a match.

During training of these tasks, half of the trials are matching and the other half are non-matching. In DMS and DNMS tasks, stimulus 1 is always drawn randomly. In half of the trials, stimulus 2 appears at the same direction as stimulus 1. In the other half, stimulus 2 is drawn randomly between 10 and 350° away from stimulus 1. In DMC and DNMC tasks, both stimulus 1 and 2 are drawn randomly and independently from the uniform distribution

$$\mathrm{U}(\{18, 54, 90, 126, 162, 198, 234, 270, 306, 342\})$$

In all Matching family tasks,

$$T_{\mathrm{delay1}} \sim \mathrm{U}(\{200, 400, 800, 1600\})$$

Also, match trials and non-match trials always appear with equal probability.

Training procedure. The loss L to be minimized is computed by time-averaging the squared errors between the network output $\mathbf{z}(t)$ and the target output $\hat{\mathbf{z}}(t)$.

$L = L_{\mathrm{mse}} \equiv \langle m_{i,t}(z_{i,t} - \hat{z}_{i,t})^2 \rangle_{i,t}$. Here, i is the index of the output units. The squared errors at different time points and of different output units are potentially weighted differently according to the non-negative mask matrix $m_{i,t}$. For the output ring units, before the response epoch, we have $m_{i,t} = 1$. The first 100 ms of the response epoch is a grace period with $m_{i,t} = 0$, while for the rest of the response epoch, $m_{i,t} = 5$. For the fixation output unit, $m_{i,t}$ is two times stronger than the mask for the output ring units.

The training is performed with Adam, a powerful variant of stochastic gradient descent[43]. We used the default set of parameters. The learning rate is 0.001, the decay rate for the first and second moment estimates are 0.9 and 0.999, respectively.

The recurrent connection matrix is initialized with a scaled identity matrix $q \cdot 1$[44], where 1 is the identity matrix. We chose $q = 0.5$ such that the gradient is roughly preserved during backpropagation when the network is initialized. The input and output connection weights are initialized as independent Gaussian random variables with mean 0, and standard deviations $1/\sqrt{N_{\mathrm{in}}}$ and $0.4/\sqrt{N_{\mathrm{rec}}}$, respectively. The standard deviation value for the output weights is chosen to prevent saturation of output units after initialization.

During training, we randomly interleaved all the tasks with equal probabilities, except for the Ctx DM 1 and Ctx DM 2 tasks that appear five times more frequently, because without sufficient training, the network gets stuck at an alternative strategy. Instead of correctly ignoring modality 1 or 2, the network can choose to ignore the context and integrate information from both modalities equally. This strategy gives the network an accuracy close to 75%. During training, we used mini-batches of 64 trials, in which all trials are generated from the same task for computational efficiency.

Task variance analysis. A central goal of our analysis was to determine whether individual units in the network are selective to different tasks, or whether units tended to be similarly selective to all tasks. To quantify how selective a unit is in one task, we defined a task variance metric. To compute the task variance $\mathrm{TV}_i(A)$ for task A and unit i, we ran the network for many stimulus conditions that span the space of possible stimuli. For example, in the DM family tasks, we ran the network for stimuli with directions ranging from 0 to 360° and with coherences ranging from almost 0 to 0.2. After running the network for many stimulus conditions, we computed the variance across stimulus conditions (trials) at each time point for a specific unit then averaged the variance across all time points to get the final task variance for this unit. The fixation epoch is excluded from this analysis. To eliminate the effect of recurrent noise, private noise to recurrent units is set to zero in this analysis. This process was repeated for each unit in the network. Therefore,

$$\mathrm{TV}_i(A) = \langle [r_i(j,t) - \langle r_i(j',t) \rangle_{j'}]^2 \rangle_{j,t}$$

where $r_i(j,t)$ is the activity of unit i on time t of trial j. In Figs. 2 and 4, we only analyzed active units, defined as those that have summed task variance across tasks higher than a threshold, 10^{-3}. The results do not depend strongly on the choice of the threshold. This procedure prevents units with extremely low task variance from being included in the analysis.

By computing each unit's selectivity across different stimulus conditions, we naturally include the selectivity to motor outputs, because motor outputs depend ultimately on the stimuli. A unit that is only selective to motor outputs or other cognitive variables in a task will still have a non-zero task variance. Units that are purely selective to rules and/or time will, however, have zero task variance and therefore be excluded from our analysis.

The clustering of units based on their task variance patterns in Fig. 2 uses k-means clustering from the Python package scikit-learn. To assess how well a clustering configuration is, we computed its silhouette score on the basis of intracluster and intercluster distances. The silhouette score of an unit i is $1 - d_{i,\mathrm{intra}}/d_{i,\mathrm{inter}}$ (assuming $d_{i,\mathrm{intra}} < d_{i,\mathrm{inter}}$), where $d_{i,\mathrm{intra}}$ is the average distance of this

unit with other units in the same cluster, and $d_{i,\text{inter}}$ is average distance between this unit and units in the nearest cluster. The silhouette score of a clustering scheme is the average silhouette score of all units. A higher silhouette score means a better clustering. We computed the silhouette for the number of clusters ranging from 2 to 30. The optimal number of clusters $\tilde{k}$ is determined by choosing the k with the highest silhouette score.

In Fig. 2d, we visualize the clustering using tSNE. For each unit, the normalized task variances across all tasks form a 20-dimensional vector that is then embedded in a two-dimensional space. For the tSNE method, we used the exact method for gradient calculation, a learning rate of 100 and a perplexity of 30.

The FTV with respect to tasks A and B is

$$\text{FTV}_i(A,B) = \frac{\text{TV}_i(A) - \text{TV}_i(B)}{\text{TV}_i(A) + \text{TV}_i(B)}$$

To obtain a statistical baseline for the FTV distributions as in Supplementary Fig. 6, we transform the neural activities of the network with a random orthogonal matrix before computing the task variance. For each network, we generate a random orthogonal matrix M using the Python package Scipy. All network activities are multiplied by this matrix M to obtain a rotated version of the original neural representation.

$$\boldsymbol{r}_t^{\text{rot}} = M\boldsymbol{r}_t$$

Since multiplying neural activities by an orthogonal matrix is equivalent to rotating and reflecting the neural representation in state space, this procedure will preserve results from state-space analysis. We then compute task variances and FTV using the rotated neural activities. The FTV distributions using the rotated activities are clearly different from the original FTV distributions.

Varying hyperparameters of neural networks. In Fig. 3, we trained networks with the following possible hyperparameters. The activation functions $f(\cdot)$ can be the Softplus function

$$f(x) = \log(1+\exp(x))$$

the ReLU

$$f(x) = \max(x, 0)$$

the Tanh function

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

and the ReTanh

$$f(x) = \max(\tanh(x), 0)$$

The network architecture can be the leaky RNN architecture defined above, or the leaky GRU architecture[42]. The leaky GRU architecture is modified on the basis of the original GRU architecture such that the network can be considered as a discretized version of a time-continuous system.

$$\begin{aligned}
\boldsymbol{\lambda}_t &= \text{sigmoid}(W^{\text{rec},\lambda}\mathbf{r}_{t-1} + W^{\text{in},\lambda}\mathbf{u}_t + \mathbf{b}^{\lambda}) \\
\boldsymbol{\gamma}_t &= \text{sigmoid}(W^{\text{rec},\gamma}\mathbf{r}_{t-1} + W^{\text{in},\gamma}\mathbf{u}_t + \mathbf{b}^{\gamma}) \\
\mathbf{r}_t &= (1-\alpha\boldsymbol{\lambda}_t)\odot\boldsymbol{r}_{t-1} + \alpha\boldsymbol{\lambda}_t \\
&\quad \cdot f(W^{\text{rec}}(\boldsymbol{\gamma}_t\odot\mathbf{r}_{t-1}) + W^{\text{in}}\mathbf{u}_t + \boldsymbol{b} + \sqrt{2\alpha^{-1}\sigma_{\text{rec}}^2}\mathbf{N}(0,1))
\end{aligned}$$

Here, $\alpha = \Delta_t/\tau$ and $\tau/\boldsymbol{\lambda}_t$ are the effective time constants of each unit. $\boldsymbol{\gamma}_t$ are the gating variables, determining the extent of which the activity of an unit is used to update the activity of other units.

The recurrent connection matrix W^{rec} is initialized either with a diagonal matrix (a scaled identity matrix) or with a random orthogonal matrix. The $N_{\text{rec}}\times N_{\text{rec}}$ random orthogonal matrix is sampled from $O(N_{\text{rec}})$, the orthogonal group in dimension N_{rec} using the Python function scipy.stats.ortho_group.

We considered L1 regularizations on rates and weights, respectively. The L1 regularization on rates is

$$L_{\text{L1,rate}} = \beta_{\text{L1,rate}}\frac{1}{N_{\text{rec}}}\sum_{i,t}|r_{i,t}|$$

We used $\beta_{\text{L1,rate}} = 0, 10^{-5}, 10^{-4}, 10^{-3}$. The L1 regularization on weights is

$$L_{\text{L1,weight}} = \beta_{\text{L1,weight}}\sum_{\eta}\frac{1}{N_\eta}\sum_{i,j}|W_{i,j}^{\eta}|$$

The sum over η is taken over all connection weights in the network, including input, recurrent and output weights. We used $\beta_{\text{L1,weight}} = 0, 10^{-5}, 10^{-4}, 10^{-3}$.

In total, 256(=2×2×2×4×4) networks are trained. None of the leaky RNN networks with Tanh and ReTanh activation functions learned to perform all 20 tasks.

Analysis of the Ctx DM 1 and 2 tasks. Group 1, 2 and 12 units in Fig. 5 are defined as those units that have FTV(Ctx DM 1, Ctx DM 2) larger than 0.9, smaller than 0.1, and in between 0.4 and 0.6. In Fig. 5e, we did not directly plot the average connection weights between groups, because that would include many connections from units with different input preferences. So we only analyzed connections between units with similar input preferences. The input preference of an unit is defined as the direction of inputs that sends the strongest modality 1 and 2 summed projection. Two units are defined to have similar input preferences if the distance between their preferred directions is less than $\pi/6$. The notched box-and-whisker plots in Fig. 5f and elsewhere showed the medians (line), the confidence interval of the median (notch) estimated through bootstrapping, the lower and upper quartile of the distribution (box) and the range of the data (whisker). These plots are generated with the Python function matplotlib.pyplot.boxplot.

State-space analysis. To compute the representation of a task in the state space, we first computed the neural activities across all possible stimulus conditions, then we averaged across all these conditions. For simplicity of the analysis, we chose to analyze only the steady-state responses during the stimulus epoch. We do so by focusing on the last time point of the stimulus epoch, $t_{\text{stim1,end}}$. So the representation of task A is

$$\tilde{\boldsymbol{r}} = \left\langle \boldsymbol{r}(j, t_{\text{stim1,end}}) \right\rangle_j$$

where $\mathbf{r}(j,t)$ is the vector of network activities at trial j and time t during task A.

For each set of tasks, we performed principal component analysis to get the lower dimensional representation. We repeated this process for different networks. The representations of each set of tasks are close to four vertices of a square. As a result, the top two principal components have similar eigenvalues and are therefore interchangeable. To better compare across networks in Fig. 6b–e, we allowed a rotation and a reflection in the space spanned by the top two PCs. For each network, the rPCs are chosen such that the Go/Ctx Dly DM 1/DMS task representation lies on the positive part of the x axis, and the Dly Go/Ctx DM 1/DNMS task lies below the x axis. The rPCs are still principal components.

Training based on pre-trained networks. In Fig. 7d,e, we pre-trained networks on one of the following two sets of tasks. Set A includes Go, Dly Go and Anti, while set B includes Ctx DM 1, Ctx DM 2, Ctx Dly DM 2. We pre-trained 20 networks for each set. Each network contains 128 ReLU units. Other hyperparameters are the same as the reference setting. After pre-training, all networks reached at least 97% accuracy on the trained set of tasks.

Following pre-training, we trained these networks on either the Dly Anti task or the Ctx Dly DM 1 task. In Fig. 7d, all connection weights and biased are trained. In Fig. 7e, only the connection weights from rule input units to recurrent units are trained.

Sequential training and continual learning. For Fig. 8, tasks appear sequentially. Each task is trained for 400,000 trials. To eliminate bias toward one modality, DM 1 and DM 2 are still trained together and interleaved, and so are Ctx DM 1 and Ctx DM 2.

Connection weights of networks are all initialized with the random orthogonal initialization described previously. We added a regularizer that protects old tasks by setting another penalty for deviations of important synaptic weights (or other parameters)[35]. When training the μth task, the regularizer is

$$L_{\text{cont}} = c_{\text{cont}}\sum_k \Omega_k^{\mu}(\theta_k - \tilde{\theta}_k)^2$$

Here, c_{cont} is the overall strength of the regularizer, θ_k denotes the kth parameter of the network. The value of the anchor parameter $\tilde{\theta}_k$ is the value of θ_k at the end of the last task (the μth task). No regularizer is used when training the first task. Also Ω_k^{μ} measures how important the parameter is. Notice that two recent proposals[34,35] for continual learning both use regularizers of this form. The two proposals differ only in how the synaptic importances are computed. We chose the method of Zenke et al.[35], because the method by Kirkpatrick et al.[34] measures the synaptic importance locally in the parameter space, resulting in underestimated and inaccurate synaptic importance values for our settings. In Zenke et al., the importance of one parameter is determined using this parameter's historic contribution to the change in the loss function. For the kth parameter, the contribution to the change in loss during task μ is

$$\omega_k^{\mu} = \sum_{t=t^{\mu-1}}^{t^{\mu}} g_k(\theta(t))\Delta\theta_k(t)$$

where $g_k(\theta(t))$ is the gradient of loss with respect to θ_k evaluated at $\theta_k(t)$, that is, $\frac{\partial L}{\partial \theta_k}|_{\theta_k(t)}$, and $\Delta\theta_k(t)$ is the parameter change taken at step t. Therefore, ω_k^μ tracks how parameter θ_k contributes to changes in the loss during the μth task (from $t^{\mu-1}$ to t^μ). The final synaptic importance is computed by first normalizing ω_k^μ with the total change in the synaptic weight $\Delta_k^\mu = \theta_k(t^\mu) - \theta_k(t^{\mu-1})$ and summing ω_k^ν for all tasks $\nu < \mu$.

$$\Omega_k^\mu = \sum_{\nu<\mu} \frac{\omega_k^\nu}{(\Delta_k^\nu)^2 + \xi}$$

The extra hyperparameter ξ prevents Ω_k^μ from becoming too large. The hyperparameters $c = 1.0$ and $\xi = 0.01$ are determined by a coarse grid search. The final loss is the sum of the squared-error loss and the continual-learning regularizer.

$$L = L_{\text{mse}} + L_{\text{cont}}$$

Even with the help of the continual-learning technique, we had difficulties training the network using our original task setups. So we made the DM tasks easier by increasing the coherences by 10 times. In addition, we used the rectified linear function as the neuronal nonlinearity, namely $f(x) = \max(x,0)$. We found that networks using rectified linear units learned context-dependent tasks (Ctx DM 1, Ctx DM 2) more easily.

Experimental data analysis. We analyzed data from two monkeys performing context-dependent DM tasks[11]. We focused on neural activities from the stimulus presentation period. Before computing the task variance using the same method described above, we first computed the trial-averaged firing rate of each unit in each task condition. For each unit, the firing rate in each trial is obtained by convolving the spikes with a Gaussian kernel of 40 ms width. For each task, we define four task conditions based on the signs of the motion and color coherence: (positive motion, positive color), (positive motion, negative color), (negative motion, positive color), (negative motion, negative color). Then we averaged the firing rate across all trials in each condition. This leaves us with four firing rate traces for each unit in each task. Then we computed the task variance for each unit in each task by calculating the variance across task conditions at every time point, then averaging across time.

It was necessary to reduce the number of task conditions to 4 from the original 36, otherwise the task variance estimates would be too noisy. We assessed how noisy the task variance estimates are by computing the task variance on the same data where the trial identities are shuffled. If there is little noise, then the task variance on the shuffled data should be close to zero.

Statistics and study design. In all boxplots, the confidence interval over the median is obtained with bootstrapping 10,000 times. No assumption was made about the data distribution.

No statistical methods were used to pre-determine sample sizes but our sample sizes are larger than those reported in previous publications[17,18]. Independently trained networks all have different random seeds for network initialization and training samples. Networks with different hyperparameters are trained using the same random seed. Data collection and analysis were not performed blind to the conditions of the experiments. As mentioned above, In Figs. 2 and 4, we exclude units with summed task variance across tasks lower than a threshold, 10^{-3}. Units with low task variance are mainly driven by injected noise, and therefore are irrelevant for our study. In Fig. 8, we exclude networks that achieved less than 80% accuracy on Ctx DM 1 and 2, because we are interested in networks that are able to perform selective integration in Ctx DM 1 and 2. A network can reach 75% accuracy even if it completely ignores the context and integrates from the two modalities equally. See the Life Sciences Reporting Summary for more information on the study design.

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

Code availability

All training and analysis codes are available on GitHub (https://github.com/gyyang/multitask).

Data availability

We provide data files in Python and MATLAB readable formats for all trained models for further analyses on Github (https://github.com/gyyang/multitask).

References

43. Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *ICLR* (2015)..
44. Le, Q. V., Jaitly, N. & Hinton, G. E. A simple way to initialize recurrent networks of rectified linear units. Preprint at *arXiv* https://arxiv.org/abs/1504.00941 (2015).

nature research

Corresponding author(s): Xiao-Jing Wang

Reporting Summary

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

Statistical parameters

When statistical analyses are reported, confirm that the following items are present in the relevant location (e.g. figure legend, table legend, main text, or Methods section).

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

Our web collection on statistics for biologists may be useful.

Software and code

Policy information about availability of computer code

Data collection	Latest versions of Python, Numpy, Scipy, and Tensorflow. Code will be made publicly available on Github (https://github.com/gyyang/multitask) upon acceptance.
Data analysis	Latest versions of Python, Numpy, Scipy, and Tensorflow. Code will be made publicly available on Github (https://github.com/gyyang/multitask) upon acceptance.

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

Data

Policy information about availability of data

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

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

All training and analysis codes will be available at publication on GitHub (https://github.com/gyyang/multitask). We will also provide data files in Python and Matlab

readable formats for all trained models for further analyses. The pretrained model will be stored in a Google Drive folder with its link provided on the same Github repository.

Field-specific reporting

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

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

For a reference copy of the document with all sections, see nature.com/authors/policies/ReportingSummary-flat.pdf

Life sciences study design

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

Sample size	No statistical methods were used to pre-determine sample sizes but our sample sizes are larger than those reported in previous publications
Data exclusions	In Figs. 2, 4, we exclude units with summed task variance across tasks lower than a threshold, 1e-3. Units with low task variance are mainly driven by injected noise, therefore are irrelevant for our study. In Fig. 8, we exclude networks that achieved less than 80% accuracy on Ctx DM 1 and 2, because we are interested in networks that are able to perform selective integration in Ctx DM 1 and 2. A network can reach 75% accuracy even if it completely ignores the context and integrates from the two modalities equally. Exclusion criteria are pre-established.
Replication	We have retrained all of our networks several times, and we are able to reproduce our primary conclusions.
Randomization	Independently trained networks all have different random seeds for network initialization and training samples. Networks with different hyperparameters are trained using the same random seed.
Blinding	Data collection and analysis were not performed blind to the conditions of the experiments

Reporting for specific materials, systems and methods

Materials & experimental systems

n/a	Involved in the study
☒	☐ Unique biological materials
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology
☒	☐ Animals and other organisms
☒	☐ Human research participants

Methods

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