nature neuroscience

Article https://doi.org/10.1038/s41593-024-01607-5

Natural language instructions induce compositional generalization in networks of neurons

Received: 13 May 2023

Accepted: 15 February 2024

Published online: 18 March 2024

Check for updates

Reidar Riveland & Alexandre Pouget

A fundamental human cognitive feat is to interpret linguistic instructions in order to perform novel tasks without explicit task experience. Yet, the neural computations that might be used to accomplish this remain poorly understood. We use advances in natural language processing to create a neural model of generalization based on linguistic instructions. Models are trained on a set of common psychophysical tasks, and receive instructions embedded by a pretrained language model. Our best models can perform a previously unseen task with an average performance of 83% correct based solely on linguistic instructions (that is, zero-shot learning). We found that language scaffolds sensorimotor representations such that activity for interrelated tasks shares a common geometry with the semantic representations of instructions, allowing language to cue the proper composition of practiced skills in unseen settings. We show how this model generates a linguistic description of a novel task it has identified using only motor feedback, which can subsequently guide a partner model to perform the task. Our models offer several experimentally testable predictions outlining how linguistic information must be represented to facilitate flexible and general cognition in the human brain.

In a laboratory setting, animals require numerous trials in order to acquire a new behavioral task. This is in part because the only means of communication with nonlinguistic animals is simple positive and negative reinforcement signals. By contrast, it is common to give written or verbal instructions to humans, which allows them to perform new tasks relatively quickly. Further, once humans have learned a task, they can typically describe the solution with natural language. The dual ability to use an instruction to perform a novel task and, conversely, produce a linguistic description of the demands of a task once it has been learned are two unique cornerstones of human communication. Yet, the computational principles that underlie these abilities remain poorly understood.

One influential systems-level explanation posits that flexible interregional connectivity in the prefrontal cortex allows for the reuse of practiced sensorimotor representations in novel settings[1,2]. More recently, multiple studies have observed that when subjects are required to flexibly recruit different stimulus-response patterns, neural representations are organized according to the abstract structure of the task set[3–5]. Lastly, recent modeling work has shown that a multitasking recurrent neural network (RNN) will share dynamical motifs across tasks with similar demands[6]. This work forms a strong basis for explanations of flexible cognition in humans but leaves open the question of how linguistic information can reconfigure a sensorimotor network so that it performs a novel task well on the first attempt. Overall, it remains unclear what representational structure we should expect from brain areas that are responsible for integrating linguistic information in order to reorganize sensorimotor mappings on the fly.

These questions become all the more pressing given that recent advances in machine learning have led to artificial systems that exhibit

Department of Basic Neuroscience, University of Geneva, Geneva, Switzerland. e-mail: reidar.riveland@unige.ch

human-like language skills[7,8]. Recent works have matched neural data recorded during passive listening and reading tasks to activations in autoregressive language models (that is, GPT[9]), arguing that there is a fundamentally predictive component to language comprehension[10,11]. Additionally, some high-profile machine learning models do show the ability to use natural language as a prompt to perform a linguistic task or render an image, but the outputs of these models are difficult to interpret in terms of a sensorimotor mapping that we might expect to occur in a biological system[12–14]. Alternatively, recent work on multimodal interactive agents may be more interpretable in terms of the actions they take, but utilize a perceptual hierarchy that fuses vision and language at early stages of processing, making them difficult to map onto functionally and anatomically distinct language and vision areas in human brains[15–17].

We, therefore, seek to leverage the power of language models in a way that results in testable neural predictions detailing how the human brain processes natural language in order to generalize across sensorimotor tasks.

To that end, we train an RNN (sensorimotor-RNN) model on a set of simple psychophysical tasks where models process instructions for each task using a pretrained language model. We find that embedding instructions with models tuned to sentence-level semantics allow sensorimotor-RNNs to perform a novel task at 83% correct, on average. Generalization in our models is supported by a representational geometry that captures task subcomponents and is shared between instruction embeddings and sensorimotor activity, thereby allowing a composition of practice skills in a novel setting. We also find that individual neurons modulate their tuning based on the semantics of instructions. We demonstrate how a network trained to interpret linguistic instructions can invert this understanding and produce a linguistic description of a previously unseen task based on the information in motor feedback signals. We end by discussing how these results can guide research on the neural basis of language-based generalization in the human brain.

Results

Instructed models and task set

We train sensorimotor-RNNs on a set of 50 interrelated psychophysical tasks that require various cognitive capacities that are well studied in the literature[18]. Two example tasks are presented in Fig. 1a,b as they might appear in a laboratory setting. For all tasks, models receive a sensory input and task-identifying information and must output motor response activity (Fig. 1c). Input stimuli are encoded by two one-dimensional maps of neurons, each representing a different input modality, with periodic Gaussian tuning curves to angles (over $(0, 2\pi)$). Output responses are encoded in the same way. Inputs also include a single fixation unit. After the input fixation is off, the model can respond to the input stimuli. Our 50 tasks are roughly divided into 5 groups, 'Go', 'Decision-making', 'Comparison', 'Duration' And 'Matching', where within-group tasks share similar sensory input structures but may require divergent responses. For instance, in the decision-making (DM) task, the network must respond in the direction of the stimulus with the highest contrast, whereas in the anti-decision-making (AntiDM) task, the network responds to the stimulus with the weakest contrast (Fig. 1a). Thus, networks must properly infer the task demands for a given trial from task-identifying information in order to perform all tasks simultaneously (see Methods for task details; see Supplementary Fig. 13 for example trials of all tasks).

In our models, task-identifying input is either nonlinguistic or linguistic. We use two nonlinguistic control models. First, in SIMPLENET, the identity of a task is represented by one of 50 orthogonal rule vectors. Second, STRUCTURENET uses a set of 10 orthogonal structure vectors, each representing a dimension of the task set (that is, respond weakest versus strongest direction), and tasks are encoded using combinations of these vectors (see Supplementary Notes 3 for the full set of structure combinations). As a result, STRUCTURENET fully captures all the relevant relationships among tasks, whereas SIMPLENET encodes none of this structure.

Instructed models use a pretrained transformer architecture[19] to embed natural language instructions for the tasks at hand. For each task, there is a corresponding set of 20 unique instructions (15 training, 5 validation; see Supplementary Notes 2 for the full instruction set). We test various types of language models that share the same basic architecture but differ in their size and also their pretraining objective. We tested two autoregressive models, a standard and a large version of GPT2, which we call GPT and GPT (XL), respectively. Previous work has demonstrated that GPT activations can account for various neural signatures of reading and listening[11]. BERT is trained to identify masked words within a piece of text[20], but it also uses an unsupervised sentence-level objective, in which the network is given two sentences and must determine whether they follow each other in the original text. SBERT is trained like BERT but receives additional tuning on the Stanford Natural Language Inference task, a hand-labeled dataset detailing the logical relationship between two candidate sentences (Methods)[21,22]. Lastly, we use the language embedder from CLIP, a multimodal model that learns a joint embedding space of images and text captions[23]. We call a sensorimotor-RNN using a given language model LANGUAGEMODELNET and append a letter indicating its size. The various sizes of models are given in Fig. 1c. For each language model, we apply a pooling method to the last hidden state of the transformer and pass this fixed-length representation through a set of linear weights that are trained during task learning. This results in a 64-dimensional instruction embedding across all models (Methods). Language model weights are frozen unless otherwise specified. Finally, as a control, we also test a bag-of-words (BoW) embedding scheme that only uses word count statistics to embed each instruction.

First, we verify our models can perform all tasks simultaneously. For instructed models to perform well, they must infer the common semantic content between 15 distinct instruction formulations for each task. We find that all our instructed models can learn all tasks simultaneously except for GPTNET, where performance asymptotes are below the 95% threshold for some tasks. Hence, we relax the performance threshold to 85% for models that use GPT (Supplementary Fig. 1; see Methods for training details). We additionally tested all architectures on validation instructions (Supplementary Fig. 2). SBERTNET (L) and SBERTNET are our best-performing models, achieving an average performance of 97% and 94%, respectively, on validation instructions, demonstrating that these networks infer the proper semantic content even for entirely novel instructions.

Generalization to novel tasks

We next examined the extent to which different language models aided generalization to novel tasks. We trained individual networks on 45 tasks and then tested performance when exposed to the five held-out tasks. We use unequal-variance t-tests to make comparisons among the performance of different models. Figure 2 shows results with P values for the most relevant comparisons (a full matrix of comparisons across all models can be found in Supplementary Figs. 3 and 4)

Our uninstructed control model SIMPLENET performs at 39%, on average, on the first presentation of a novel task (zero-shot generalization). This serves as a baseline for generalization. Note that despite the orthogonality of task rules provided to SIMPLENET, exposure to the task set allows models to learn patterns that are common to all tasks (for example, always repress response during fixation). Therefore, 39% is not chance-level performance per se, but rather performance achieved by a network trained and tested on a task set with some common requirements for responding. GPTNET, exhibits a zero-shot generalization of 57%. This is a significant improvement over SIMPLENET ($t = 8.32$, $P = 8.24 \times 10^{-16}$). Strikingly, increasing the size of GPT by an order of magnitude to the 1.5 billion parameters used by GPT (XL) only resulted

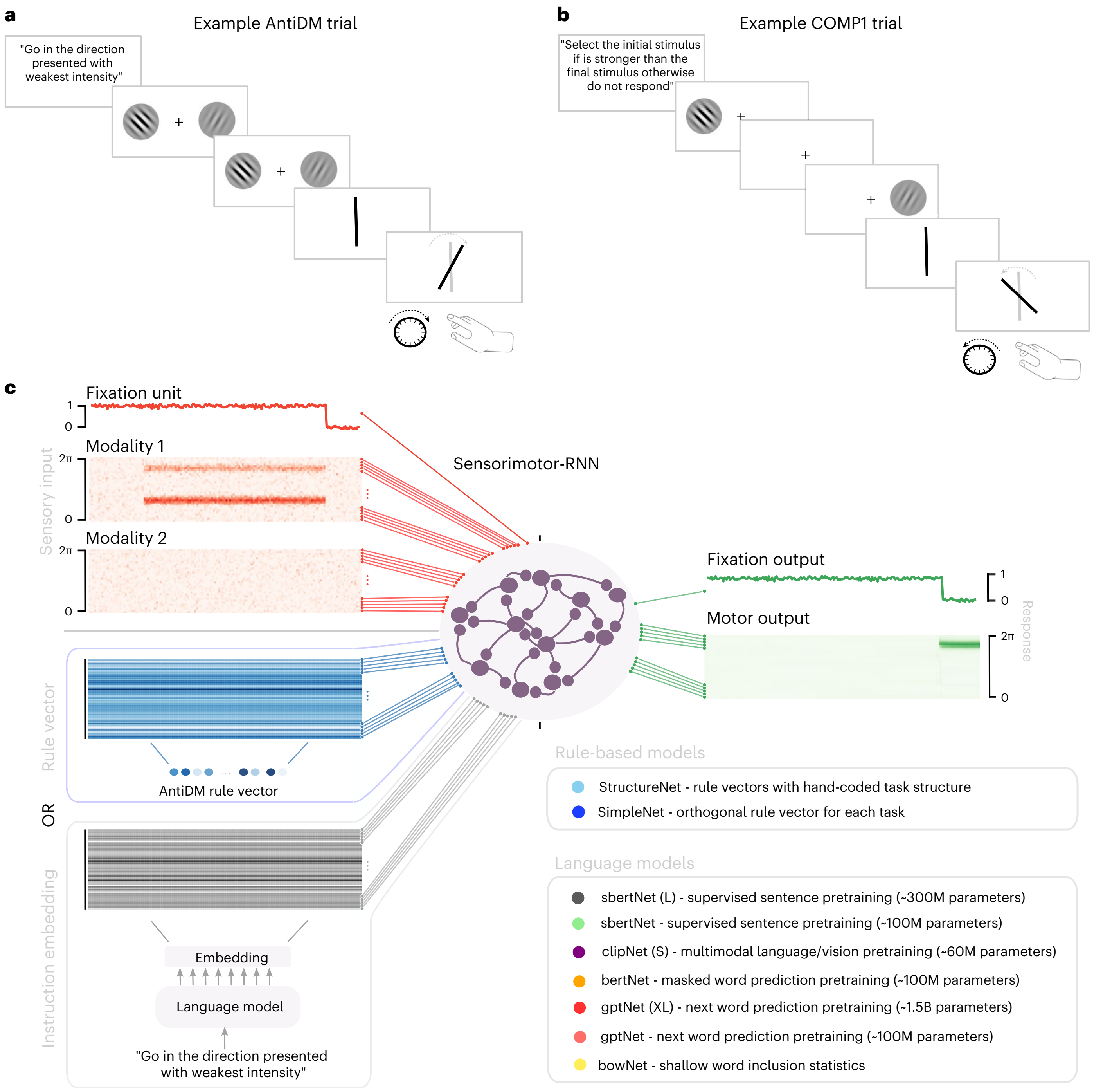

a
Example AntiDM trial
"Go in the direction presented with weakest intensity"
b
Example COMP1 trial
"Select the initial stimulus if is stronger than the final stimulus otherwise do not respond"
c
Fixation unit
Modality 1
Modality 2
Sensory input
Sensorimotor-RNN
Fixation output
Motor output
Response
Rule vector
AntiDM rule vector
OR
Instruction embedding
Embedding
Language model
"Go in the direction presented with weakest intensity"
Rule-based models
StructureNet - rule vectors with hand-coded task structure
SimpleNet - orthogonal rule vector for each task
Language models
sbertNet (L) - supervised sentence pretraining (~300M parameters)
sbertNet - supervised sentence pretraining (~100M parameters)
clipNet (S) - multimodal language/vision pretraining (~60M parameters)
bertNet - masked word prediction pretraining (~100M parameters)
gptNet (XL) - next word prediction pretraining (~1.5B parameters)
gptNet - next word prediction pretraining (~100M parameters)
bowNet - shallow word inclusion statistics

Fig. 1 | Tasks and models. a,**b**, Illustrations of example trials as they might appear in a laboratory setting. The trial is instructed, then stimuli are presented with different angles and strengths of contrast. The agent must then respond with the proper angle during the response period. **a**, An example AntiDM trial where the agent must respond to the angle presented with the least intensity. **b**, An example COMP1 trial where the agent must respond to the first angle if it is presented with higher intensity than the second angle otherwise repress response. **c**, Diagram of model inputs and outputs. Sensory inputs (fixation unit, modality 1, modality 2) are shown in red and model outputs (fixation output, motor output) are shown in green. Models also receive a rule vector (blue) or the embedding that results from passing task instructions through a pretrained language model (gray). A list of models tested is provided in the inset.

in modest gains over BOWNET (64%), with GPTNET (XL) achieving 68% on held-out tasks ($t = 2.04$, $P = 0.047$). By contrast, CLIPNET (S), which uses 4% of the number of parameters utilized by GPTNET (XL), is nonetheless able to achieve the same performance (68% correct, $t = 0.146$, $P = 0.88$). Likewise, BERTNET achieves a generalization performance that lags only 2% behind GPTNETXL in the mean ($t = -1.122$, $P = 0.262$). By contrast, models with knowledge of sentence-level semantics show marked improvements in generalization, with SBERTNET performing an unseen task at 79% correct on average. Finally, our best-performing model, SBERTNET (L), can execute a never-before-seen task with a performance of 83% correct, on average, lagging just a few percentage points behind STRUCTURENET (88% correct), which receives the structure of the task set hand-coded in its rule vectors.

Figure 2b shows a histogram of the number of tasks for which each model achieves a given level of performance. Again, SBERTNET (L) manages to perform over 20 tasks set nearly perfectly in the zero-shot

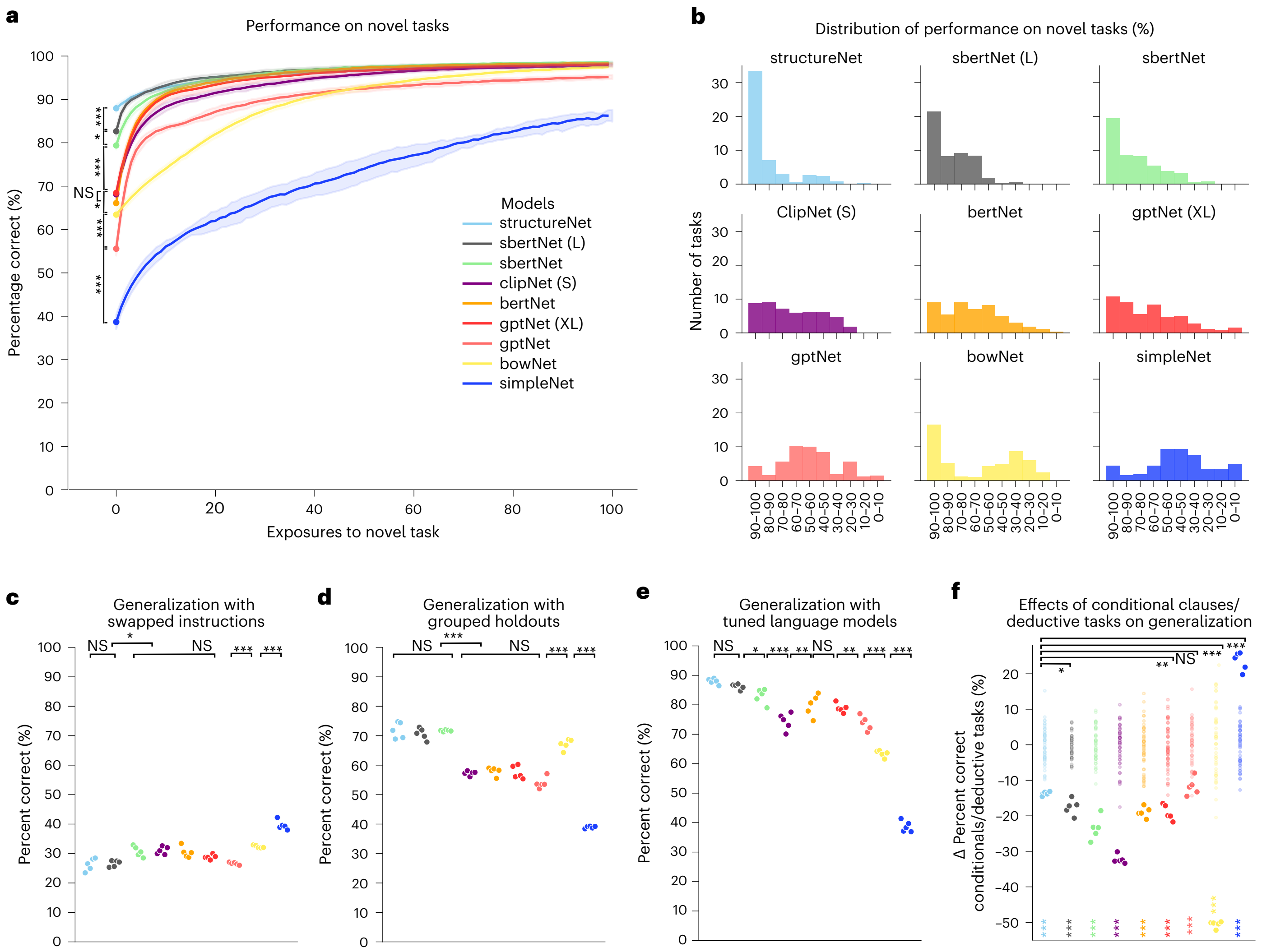


Fig. 2 | Model performance on novel tasks. **a**, Learning curves for the first 100 exposures to held-out tasks averaged over all tasks. Data are presented as the mean ± s.d. across different $n = 5$ random initializations of sensorimotor-RNN weights. For all subplots, asterisks indicate significant differences among performance according to a two-sided unequal-variance t-test. Most relevant comparisons are presented in plots (for all subplots, not significant (NS), $P > 0.05$, $^{*}P < 0.05$, $^{**}P < 0.01$, $^{***}P < 0.001$; STRUCTURENET versus SBERTNET (L): $t = 3.761$, $P = 1.89 \times 10^{-4}$; SBERTNET (L) versus SBERTNET: $t = 2.19$, $P = 0.029$; SBERTNET versus CLIPNET: $t = 6.22$, $P = 1.02 \times 10^{-9}$; CLIPNET versus BERTNET: $t = 1.037$, $P = 0.300$; BERTNET versus GPTNET (XL): $t = -1.122$, $P = 0.262$; GPTNET (XL) versus GPTNET: $t = 6.22$, $P = 1.04 \times 10^{-9}$; GPTNET versus BOWNET: $t = -3.346$, $P = 8.85 \times 10^{-4}$; BOWNET versus SIMPLENET: $t = 10.25$, $P = 2.091 \times 10^{-22}$). A full table of pairwise comparisons can be found in Supplementary Fig. 3. **b**, Distribution of generalization performance (that is, first exposure to novel task) across models. **c**–**f**, Performance across different test conditions for $n = 5$ different random initialization of sensorimotor-RNN weights where each point indicates average performance across tasks for a given initialization. **c**, Generalization performance for tasks where instructions are swapped at test time (STRUCTURENET versus SBERTNET (L): $t = -0.15$, $P = 0.875$; SBERTNET (L) versus SBERTNET: $t = -2.102$, $P = 0.036$; SBERTNET versus CLIPNET: $t = -0.162$, $P = 0.871$; CLIPNET versus BERTNET: $t = 0.315$, $P = 0.752$; BERTNET versus GPTNET (XL): $t = 0.781$, $P = 0.435$; GPTNET (XL) versus GPTNET: $t = 1.071$, $P = 0.285$; GPTNET versus BOWNET: $t = -2.702$, $P = 0.007$; BOWNET versus SIMPLENET: $t = -3.471$, $P = 5.633^{-4}$). A full table of pairwise comparisons can be found in Supplementary Fig. 4. **d**, Generalization performance for models where tasks from the same family are held out during training (STRUCTURENET versus SBERTNET (L): $t = 0.629$, $P = 0.530$; SBERTNET (L) versus SBERTNET: $t = -0.668$, $P = 0.504$; SBERTNET versus CLIPNET: $t = 8.043$, $P = 7.757 \times 10^{-15}$; CLIPNET versus BERTNET: $t = -0.306$, $P = 0.759$; BERTNET versus GPTNET (XL): $t = 0.163$, $P = 0.869$; GPTNET (XL) versus GPTNET: $t = 1.534$, $P = 0.126$; GPTNET versus BOWNET: $t = -6.418$, $P = 3.26 \times 10^{-10}$; BOWNET versus SIMPLENET: $t = 14.23$, $P = 8.561^{-39}$). A full table of pairwise comparisons can be found in Supplementary Fig. 4. **e**, Generalization performance for models where the last layers of language models are allowed to fine-tune to the loss from sensorimotor tasks (STRUCTURENET versus SBERTNET (L): $t = 1.203$, $P = 0.229$; SBERTNET (L) versus SBERTNET: $t = 2.399$, $P = 0.016$; SBERTNET versus CLIPNET: $t = 5.186$, $P = 3.251 \times 10^{-7}$; CLIPNET versus BERTNET: $t = -3.002$, $P = 0.002$; BERTNET versus GPTNET (XL): $t = 0.522$, $P = 0.601$; GPTNET (XL) versus GPTNET: $t = 2.631$, $P = 0.009$; GPTNET versus BOWNET: $t = 4.440$, $P = 1.134 \times 10^{-5}$; BOWNET versus SIMPLENET: $t = 10.255$, $P = 2.091 \times 10^{-22}$). A full table of pairwise comparisons can be found in Supplementary Fig. 4. **f**, Average difference in performance between tasks that use standard imperative instructions and those that use instructions with conditional clauses and require a simple deductive reasoning component. Colored asterisks at the bottom of the plot show P values for a two-sided, unequal-variance t-test between the null distribution constructed using random splits of the task set (transparent points represent mean differences for random splits; STRUCTURENET: $t = -36.46$, $P = 4.34 \times 10^{-23}$; SBERTNET (L): $t = -16.38$, $P = 3.02 \times 10^{-5}$; SBERTNET: $t = -15.35$, $P = 3.920 \times 10^{-5}$; CLIPNET: $t = -44.68$, $P = 5.32 \times 10^{-13}$; BERTNET: $t = -25.51$, $P = 3.14 \times 10^{-8}$; GPTNET (XL): $t = -16.99$, $P = 3.61 \times 10^{-6}$; GPTNET: $t = -9.150$, $P = 0.0002$; BOWNET: $t = -70.99$, $P = 4.566 \times 10^{-35}$; SIMPLENET: $t = 19.60$, $P = 5.82 \times 10^{-6}$), and asterisks at the top of plot indicate P-value results from a t-test comparing differences with STRUCTURENET and our other instructed models (versus SBERTNET (L): $t = 3.702$, $P = 0.0168$; versus SBERTNET: $t = 6.592$, $P = 0.002$; versus CLIPNET: $t = 30.35$, $P = 2.367 \times 10^{-7}$; versus BERTNET: $t = 7.234$, $P = 0.0007$; versus GPTNET (XL): $t = 5.282$, $P = 0.004$; versus GPTNET: $t = -1.745$, $P = 0.149$; versus BOWNET: $t = 75.04$, $P = 9.96 \times 10^{-11}$; versus SIMPLENET: $t = -30.95$, $P = 2.86 \times 10^{-6}$; see Methods and Supplementary Fig. 6. for full comparisons).

setting (for individual task performance for all models across tasks, see Supplementary Fig. 3).

To validate that our best-performing models leveraged the semantics of instructions, we presented the sensory input for one held-out task while providing the linguistic instructions for a different held-out task. Models that truly rely on linguistic information should be most penalized by this manipulation and, as predicted, we saw the largest decrease in performance for our best models (Fig. 2c).

We also tested a more stringent hold-out procedure where we purposefully chose 4–6 tasks from the same family of tasks to hold out during training (Fig. 2d). Overall, performance decreased in this more difficult setting, although our best-performing models still showed strong generalization, with SBERTNET (L) and SBERTNET achieving 71% and 72% correct on novel tasks, respectively, which was not significantly different from STRUCTURENET at 72% ($t = 0.629$, $P = 0.529$; $t = 0.064$, $P = 0.948$; for SBERTNET (L) and SBERTNET, respectively).

In addition, we tested models in a setting where we allow the weights of language models to tune according to the loss experienced during sensorimotor training (see Methods for tuning details). This manipulation improved the generalization performance across all models, and for our best-performing model, SBERTNET (L), we see that generalization is as strong as for STRUCTURENET (86%, $t = 1.204$, $P = 0.229$).

Following ref. 18, we tested models in a setting where task-type information for a given task was represented as a composition of information for related tasks in the training set (that is, AntiDMMod1 = (rule(AntiDMMod2) − rule(DMMod2)) + rule(DMMod1)). In this setting, we did find that the performance of SIMPLENET improved (60% correct). However, when we combined embedded instructions according to the same compositional rules, our linguistic models dramatically outperformed SIMPLENET. This suggests that training in the context of language more readily allows a simple compositional scheme to successfully configure task responses (see Supplementary Fig. 5 for full results and compositional encodings).

Finally, we tested a version of each model where outputs of language models are passed through a set of nonlinear layers, as opposed to the linear mapping used in the preceding results. We found that this manipulation reduced performance, suggesting that this added power leads to overfitting on training tasks, and that a simpler linear mapping is better suited to generalization (see Methods for details and Supplementary Fig. 4 for full results).

The discrepancy in performance between our instructed models suggests that in order to represent linguistic information such that it can successfully configure sensorimotor networks, it is not sufficient to simply use any very powerful language processing system. Rather, model success can be delineated by the extent to which they are exposed to sentence-level semantics during pretraining. Our best-performing models SBERTNET (L) and SBERTNET are explicitly trained to produce good sentence embeddings, whereas our worst-performing model, GPTNET, is only tuned to the statistics of upcoming words. Both CLIPNET (S) and BERTNET are exposed to some form of sentence-level knowledge. CLIPNET (S) is interested in sentence-level representations, but trains these representations using the statistics of corresponding vision representations. BERTNET performs a two-way classification of whether or not input sentences are adjacent in the training corpus. That the 1.5 billion parameters of GPTNET (XL) doesn't markedly improve performance relative to these comparatively small models speaks to the fact that model size isn't the determining factor. Lastly, although BoW removes key elements of linguistic meaning (that is, syntax), the simple use of word occurrences encodes information primarily about the similarities and differences between the sentences. For instance, simply representing the inclusion or exclusion of the words 'stronger' or 'weaker' is highly informative about the meaning of the instruction.

We also investigated which features of language make it difficult for our models to generalize. Thirty of our tasks require processing instructions with a conditional clause structure (for example, COMP1) as opposed to a simple imperative (for example, AntiDM). Tasks that are instructed using conditional clauses also require a simple form of deductive reasoning (if p then q else s). Neuroimaging literature exploring the relationship between such deductive processes and language areas has reached differing conclusions, with some early studies showing that deduction recruits regions that are thought to support syntactic computations[24–26] and follow-up studies claiming that deduction can be reliably dissociated from language areas[27–30]. One theory for this variation in results is that baseline tasks used to isolate deductive reasoning in earlier studies used linguistic stimuli that required only superficial processing[31,32].

To explore this issue, we calculated the average difference in performance between tasks with and without conditional clauses/deductive reasoning requirements (Fig. 2f). All our models performed worse on these tasks relative to a set of random shuffles. However, we also saw an additional effect between STRUCTURENET and our instructed models, which performed worse than STRUCTURENET by a statistically significant margin (see Supplementary Fig. 6 for full comparisons). This is a crucial comparison because STRUCTURENET performs deductive tasks without relying on language. Hence, the decrease in performance between STRUCTURENET and instructed models is in part due to the difficulty inherent in parsing syntactically more complicated language. The implication is that we may see engagement of linguistic areas in deductive reasoning tasks, but this may simply be due to the increased syntactic demands of corresponding instructions (rather than processes that recruit linguistic areas to explicitly aid in the deduction). This result largely agrees with two reviews of the deductive reasoning literature, which concluded that the effects in language areas seen in early studies were likely due to the syntactic complexity of test stimuli[31,32].

Shared structure in language and sensorimotor networks

We then turned to an investigation of the representational scheme that supports generalization. First, we note that like in other multitasking models, units in our sensorimotor-RNNs exhibited functional clustering, where similar subsets of neurons show high variance across similar sets of tasks (Supplementary Fig. 7). Moreover, we found that models can learn unseen tasks by only training sensorimotor-RNN input weights and keeping the recurrent dynamics constant (Supplementary Fig. 8). Past work has shown that these properties are characteristic of networks that can reuse the same set of underlying neural resources across different settings[6,18]. We then examined the geometry that exists between the neural representations of related tasks. We plotted the first three principal components (PCs) of sensorimotor-RNN hidden activity at stimulus onset in SIMPLENET, GPTNETXL, SBERTNET (L) and STRUCTURENET performing modality-specific DM and AntiDM tasks. Here, models receive input for a decision-making task in both modalities but must only attend to the stimuli in the modality relevant for the current task. Importantly, AntiDMMod1 is held out of training in the following examples. In addition, we plotted the PCs of either the rule vectors or the instruction embeddings in each task (Fig. 3).

For STRUCTURENET, hidden activity is factorized along task-relevant axes, namely a consistent 'Pro' versus 'Anti' direction in activity space (solid arrows), and a 'Mod1' versus 'Mod2' direction (dashed arrows). Importantly, this structure is maintained even for AntiDMMod1, which has been held out of training, allowing STRUCTURENET to achieve a performance of 92% correct on this unseen task. This factorization is also reflected in the PCs of rule embeddings. Strikingly, SBERTNET (L) also organizes its representations in a way that captures the essential compositional nature of the task set using only the structure that it has inferred from the semantics of instructions. This is the case for language embeddings, which maintain abstract

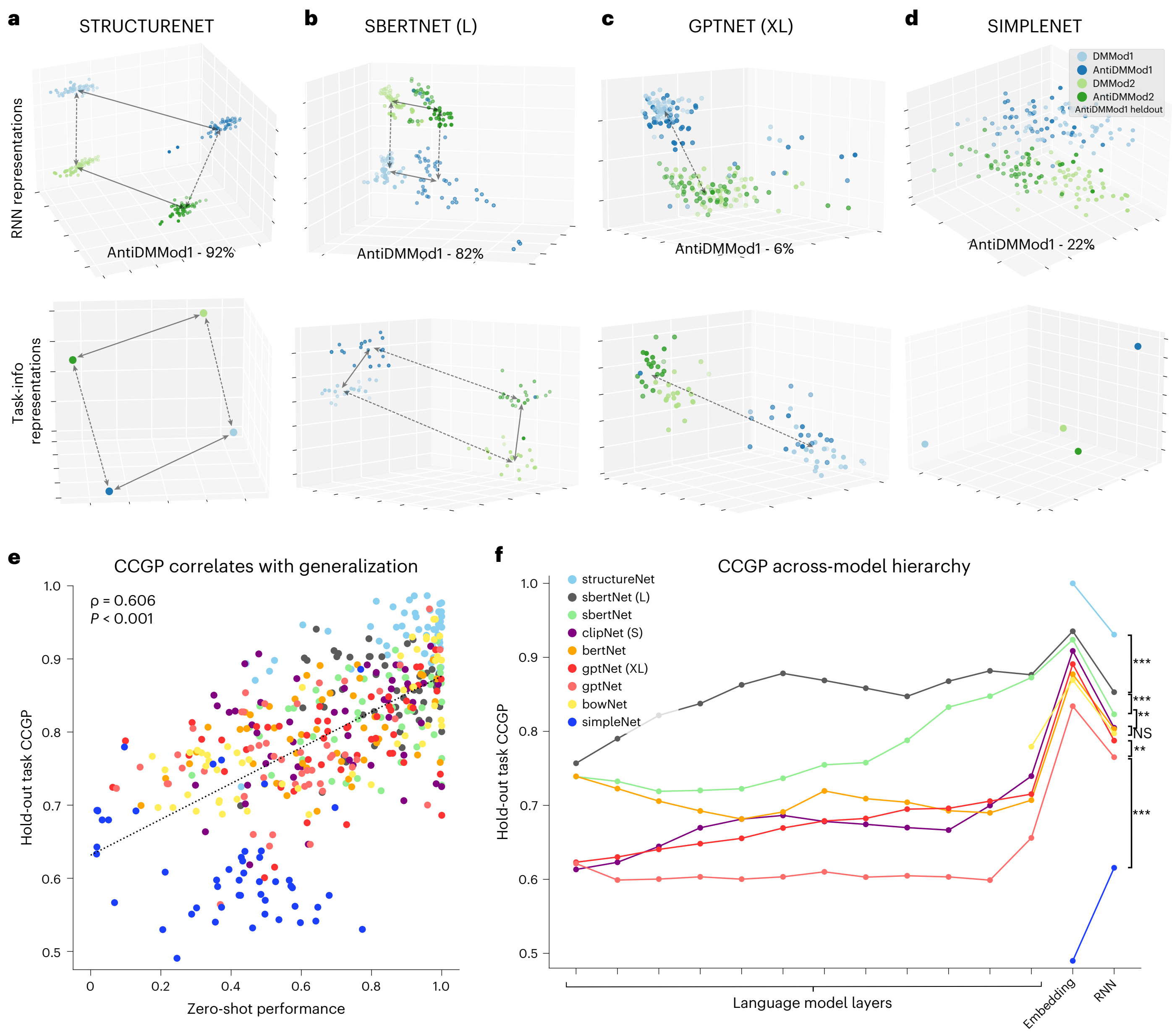


Fig. 3 | Structured representations in instructed models. a–d, The first three PCs of sensorimotor hidden activity and task-info representations for models trained with AntiDMMod1 held out. Solid arrows represent an abstract 'Pro' versus 'Anti' axis, and dashed arrows represent an abstract 'Mod1' versus 'Mod2' axis. **a**, STRUCTURENET. **b**, SBERTNET (L). **c**, GPTNET (XL). **d**, SIMPLENET. **e**, Correlation between held-out task CCGP and zero-shot performance (Pearson's $r = 0.606$, $P = 1.57 \times 10^{-46}$). **f**, CCGP scores for held-out tasks for each layer in the model hierarchy. Significance scores indicate P-value results from pairwise two-sided unequal-variance t-tests performed among model distributions of CCGP scores on held-out tasks for sensorimotor-RNN (NS $P > 0.05$, $^{*}P < 0.05$, $^{**}P < 0.01$, $^{***}P < 0.001$; STRUCTURENET versus SBERTNET (L): $t = 13.67$, $P = 2.44 \times 10^{-36}$; SBERTNET (L) versus SBERTNET: $t = 5.061$, $P = 5.84 \times 10^{-7}$; SBERTNET versus CLIPNET: $t = 2.809$, $P = 0.005$; CLIPNET versus BERTNET: $t = 0.278$, $P = 0.780$; BERTNET versus GPTNET (XL): $t = 2.505$, $P = 0.012$; GPTNET (XL) versus GPTNET: $t = 3.180$, $P = 0.001$; GPTNET versus BOWNET: $t = -4.176$, $P = 3.50 \times 10^{-5}$; BOWNET versus SIMPLENET: $t = 23.0.8$, $P = 1.10^{-80}$; see Supplementary Fig. 9 for full comparisons as well as t-test results for embedding layer CCGP scores).

axes across AntiDMMod1 instructions (again, held out of training). As a result, SBERTNET (L) is able to use these relevant axes for AntiDMMod1 sensorimotor-RNN representations, leading to a generalization performance of 82%. By contrast, GPTNET (XL) fails to properly infer a distinct 'Pro' versus 'Anti' axes in either sensorimotor-RNN representations or language embeddings leading to a zero-shot performance of 6% on AntiDMMod1 (Fig. 3b). Finally, we find that the orthogonal rule vectors used by simpleNet preclude any structure between practiced and held-out tasks, resulting in a performance of 22%.

To more precisely quantify this structure, we measure the cross-conditional generalization performance (CCGP) of these representations[3]. CCGP measures the ability of a linear decoder trained to differentiate one set of conditions (that is, DMMod2 and AntiDMMod2) to generalize to an analogous set of test conditions (that is, DMMod1 and AntiDMMod1). Intuitively, this captures the extent to which models have learned to place sensorimotor activity along abstract task axes (that is, the 'Anti' dimension). Notably, high CCGP scores and related measures have been observed in experiments that required human participants to flexibly switch between different interrelated tasks[4,33].

We measured CCGP scores among representations in sensorimotor-RNNs for tasks that have been held out of training (Methods) and found a strong correlation between CCGP scores and zero-shot performance (Fig. 3e). Additionally, we find that swapping task instructions for held-out tasks dramatically reduces CCGP scores for all our instructed models, indicating that the semantic of instructions is crucial for maintaining structured representations (Supplementary Fig. 9).

We then looked at how structure emerges in the language processing hierarchy. CCGP decoding scores for different layers in our model are shown in Fig. 3f. For each instructed model, scores for 12 transformer layers (or the last 12 layers for SBERTNET (L) and GPTNET (XL)), the 64-dimensional embedding layer and the Sensorimotor-RNN task representations are plotted. We also plotted CCGP scores for the rule embeddings used in our nonlinguistic models. Among models, there was a notable discrepancy in how abstract structure emerges. Autoregressive models (GPTNETXL, GPTNET), BERTNET and CLIPNET (S), showed a low CCGP throughout language model layers followed by a jump in the embedding layer. This is because weights feeding into the embedding layer are tuned during sensorimotor training. The implication of this spike is that most of the useful representational processing in these models actually does not occur in the pretrained language model per se, but rather in the linear readout, which is exposed to task structure via training. By contrast, our best-performing models SBERTNET and SBERTNET (L) use language representations where high CCGP scores emerge gradually in the intermediate layers of their respective language models. Because semantic representations already have such a structure, most of the compositional inference involved in generalization can occur in the comparatively powerful language processing hierarchy. As a result, representations are already well organized in the last layer of language models, and a linear readout in the embedding layer is sufficient for the sensorimotor-RNN to correctly infer the geometry of the task set and generalize well.

This analysis strongly suggests that models exhibiting generalization do so by leveraging structured semantic representations to properly relate practiced and novel tasks in sensorimotor space, thereby allowing a composition of practiced behaviors in an unseen setting.

Semantic modulation of single-unit tuning properties

Next, we examined tuning profiles of individual units in our sensorimotor-RNNs. We found that individual neurons are tuned to a variety of task-relevant variables. Critically, however, we find neurons where this tuning varies predictably within a task group and is modulated by the semantic content of instructions in a way that reflects task demands.

For instance, in the 'Go' family of tasks, unit 42 shows direction selectivity that shifts by π between 'Pro' and 'Anti' tasks, reflecting the relationship of task demands in each context (Fig. 4a). This flip in selectivity is observed even for the AntiGo task, which was held out during training.

For the 'Matching' family of tasks, unit 14 modulates activity between 'match' (DMS, DMC) and 'non-match' (DNMS, DNMC) conditions. In 'non-match' trials, the activity of this unit increases as the distance between the two stimuli increases. By contrast, for 'matching' tasks, this neuron is most active when the relative distance between the two stimuli is small. Hence, in both cases this neuron modulates its activity to represent when the model should respond, changing selectivity to reflect opposing task demands between 'match' and 'non-match' trials. This is true even for DMS, which has been held out of training.

Figure 4c shows traces of unit 3 activity in modality-specific versions of DM and AntiDM tasks (AntiDMMod1 is held out of training) for different levels of contrast (contrast = $str_{\text{stim1}} - str_{\text{stim2}}$). In all tasks, we observed ramping activity where the rate of ramping is relative to the strength of contrast. This motif of activity has been reported in previous studies[34,35]. However, in our models, we observe that an evidence-accumulating neuron can swap the sign of its integration in response to a change in linguistic instructions, which allows models to meet opposing demands of 'Pro' and 'Anti' versions of the task, even for previously unseen tasks.

Interestingly, we also found that unsuccessful models failed to properly modulate tuning preferences. For example, with GPTNET (XL), which failed to factorize along a 'Pro' versus 'Anti' axis (Fig. 3b) and had poor generalization on AntiDMMod1, we also find neurons that failed to swap their sign of integration in the held-out setting (Supplementary Fig. 10).

Finally, we see a similar pattern in the time course of activity for trials in the 'Comparison' family of tasks (Fig. 4d). In the COMP1 task, the network must respond in the direction of the first stimulus if it has higher intensity than the second stimulus, and must not respond otherwise. In COMP2, it must only respond to the second stimulus if the second stimulus is higher intensity. For 'Anti' versions, the demands of stimulus ordering are the same except the model has to choose the stimuli with the weakest contrast. Even with this added complexity, we found individual neurons that modulate their tuning with respect to task demands, even for held-out tasks (in this case COMP2). For example, unit 82 is active when the network should repress response. For 'COMP1', this unit is highly active with negative contrast (that is, $str_{\text{stim2}} > str_{\text{stim1}}$), but flips this sensitivity in COMP2 and is highly active with positive contrast (that is, $str_{\text{stim1}} > str_{\text{stim2}}$). Importantly, this relation is reversed when the goal is to select the weakest stimuli. Hence, despite these subtle syntactic differences in instruction sets, the language embedding can reverse the tuning of this unit in a task-appropriate manner.

Linguistic communication between networks

We now seek to model the complementary human ability to describe a particular sensorimotor skill with words once it has been acquired. To do this, we inverted the language-to-sensorimotor mapping our models learn during training so that they can provide a linguistic description of a task based only on the state of sensorimotor units. First, we constructed an output channel (production-RNN; Fig. 5a–c), which is trained to map sensorimotor-RNN states to input instructions. We then present the network with a series of example trials while withholding instructions for a specific task. During this phase all model weights are frozen, and models receive motor feedback in order to update the embedding layer activity in order to reduce the error of the output (Fig. 5b). Once the activity in the embedding layer drives sensorimotor units to achieve a performance criterion, we used the production-RNN to decode a linguistic description of the current task. Finally, to evaluate the quality of these instructions, we input them into a partner model and measure performance across tasks (Fig. 5c). All instructing and partner models used in this section are instances of SBERTNET (L) (Methods).

Some example decoded instructions for the AntiDMMod1 task (Fig. 5d; see Supplementary Notes 4 for all decoded instructions). To visualize decoded instructions across the task set, we plotted a confusion matrix where both sensorimotor-RNN and production-RNN are trained on all tasks (Fig. 5e). Note that many decoded instructions were entirely 'novel', that is, they were not included in the training set for the production-RNN (Methods). Novel instructions made up 53% of decoded instructions across all tasks.

To test the quality of these novel instructions, we evaluated a partner model's performance on instructions generated by the first network (Fig. 5c; results are shown in Fig. 5f). When the partner model is trained on all tasks, performance on all decoded instructions was 93% on average across tasks. Communicating instructions to partner models with tasks held out of training also resulted in good performance (78%). Importantly, performance was maintained even for 'novel' instructions, where average performance was 88% for partner models trained on all tasks and 75% for partner models with hold-out tasks. Given that the instructing and partner models share the same

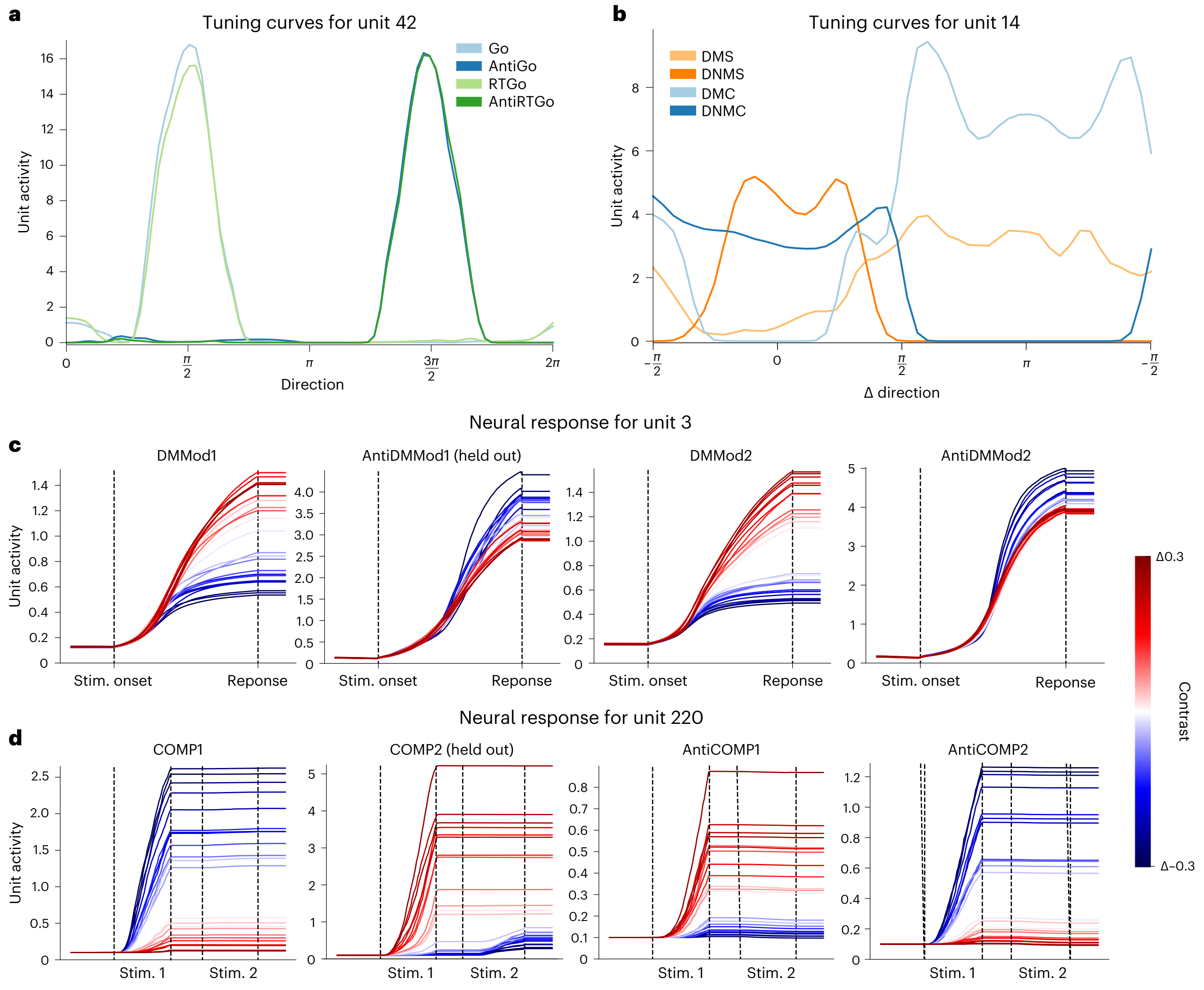


Fig. 4 | Semantic modulation of single-unit tuning properties. a, Tuning curves for a SBERTNET (L) sensorimotor-RNN unit that modulates tuning according to task demands in the 'Go' family. **b**, Tuning curves, for a SBERTNET (L) sensorimotor-RNN unit in the 'matching' family of tasks plotted in terms of difference in angle between two stimuli. **c**, Full activity traces for modality-specific 'DM' and 'AntiDM' tasks for different levels of relative stimulus strength. **d**, Full activity traces for tasks in the 'comparison' family of tasks for different levels of relative stimulus strength.

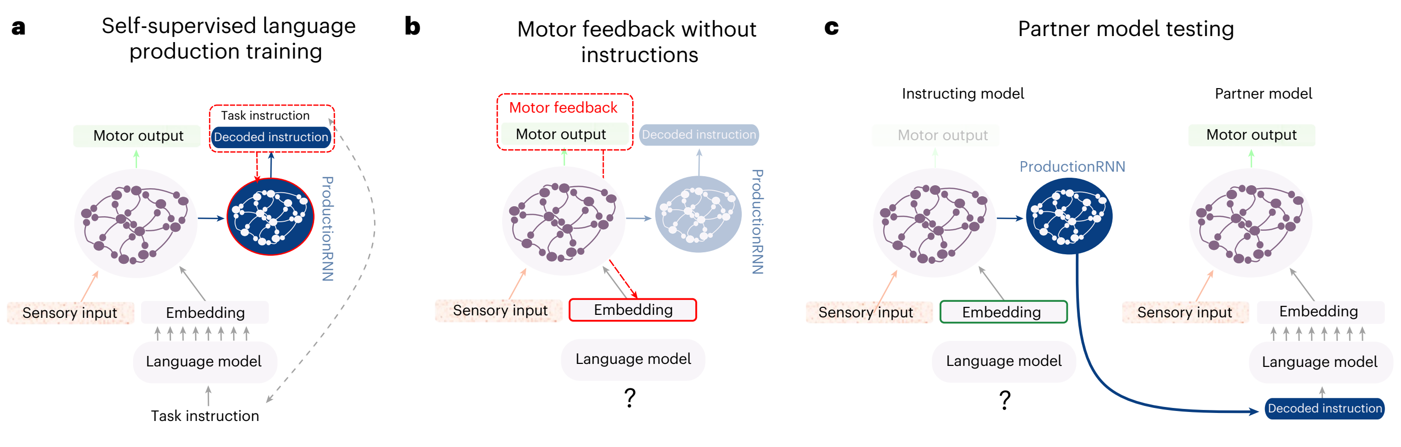
architecture, one might expect that it is more efficient to forgo the language component of communication and simply copy the embedding inferred by one model into the input of the partner model. This resulted in only 31% correct performance on average and 28% performance when testing partner models on held-out tasks. Although both instructing and partner networks share the same architecture and the same competencies, they nonetheless have different synaptic weights. Hence, using a neural representation tuned for the set of weights within the one agent won't necessarily produce good performance in the other.

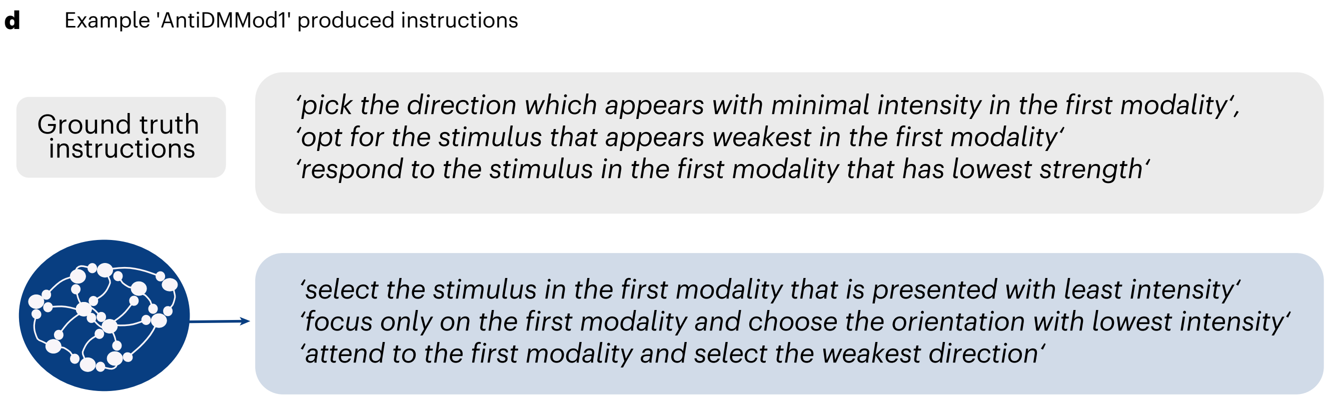
We also tested an instructing model using a sensorimotor-RNN with tasks held out of training. We emphasize here that during training the production-RNN attempts to decode from sensorimotor hidden states induced by instructions for tasks the network has never experienced before (Fig. 5a), whereas during test time, instructions are produced from sensorimotor states that emerge entirely as a result of minimizing a motor error (Fig. 5b,c). We nonetheless find that, in this setting, a partner model trained on all tasks performs at 82% correct, while partner models with tasks held out of training perform at 73%. Here, 77% of produced instructions are novel, so we see a very small decrease of 1% when we test the same partner models only on novel instructions. Like above, context representations induce a relatively low performance of 30% and 37% correct for partners trained on all tasks and with tasks held out, respectively.

Lastly, we tested our most extreme setting where tasks have been held out for both sensorimotor-RNNs and production-RNNs (Fig. 5f). We find that produced instructions induce a performance of 71% and 63% for partner models trained on all tasks and with tasks held out, respectively. Although this is a decrease in performance from our previous set-ups, the fact that models can produce sensible instructions at all in this double held-out setting is striking. The fact that the system succeeds to any extent speaks to strong inductive biases introduced by training in the context of rich, compositionally structured semantic representations.

Discussion

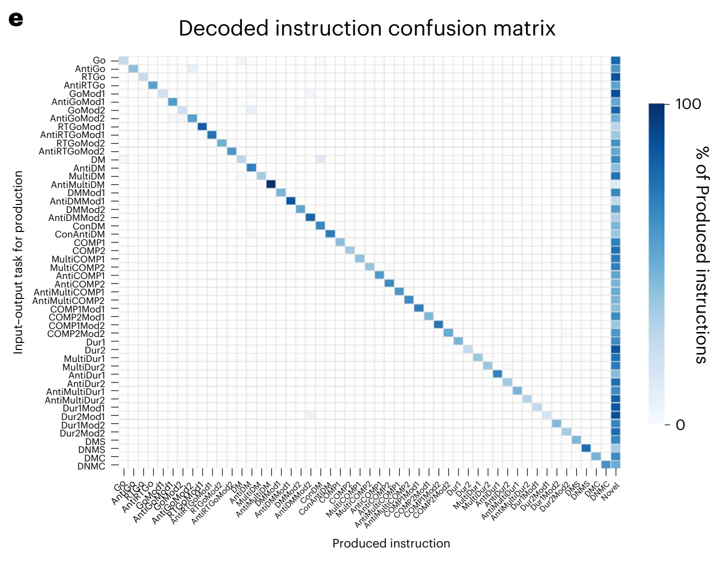
In this study, we use the latest advances in natural language processing to build tractable models of the ability to interpret instructions to




d
Example 'AntiDMMod1' produced instructions
Ground truth instructions

‘pick the direction which appears with minimal intensity in the first modality‘,
‘opt for the stimulus that appears weakest in the first modality‘
‘respond to the stimulus in the first modality that has lowest strength‘

‘select the stimulus in the first modality that is presented with least intensity‘
‘focus only on the first modality and choose the orientation with lowest intensity‘
‘attend to the first modality and select the weakest direction‘



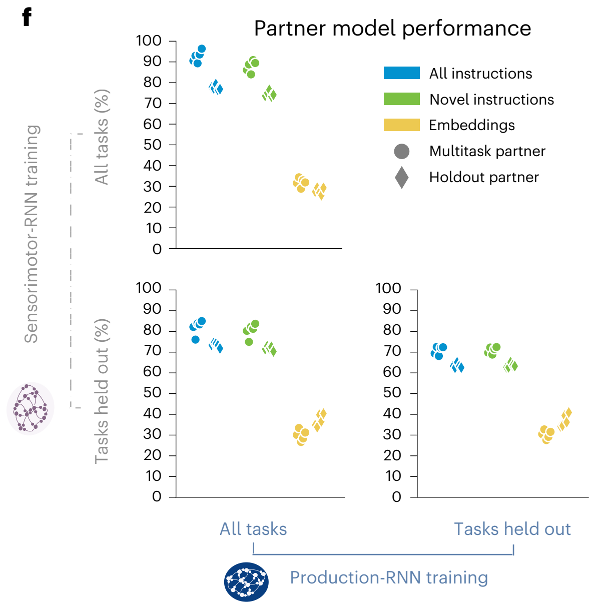


Fig. 5 | Communication between networks. a, Illustration of self-supervised training procedure for the language production network (blue). The red dashed line indicates gradient flow. **b**, Illustration of motor feedback used to drive task performance in the absence of linguistic instructions. **c**, Illustration of the partner model evaluation procedure used to evaluate the quality of instructions generated from the instructing model. **d**, Three example instructions produced from sensorimotor activity evoked by embeddings inferred in **b** for an AntiDMMod1 task. **e**, Confusion matrix of instructions produced again using the method described in **b**. *y* axis indicates input–output task used to infer an embedding, and *x* axis indicates whether the instruction produced from the resulting sensorimotor activity was included in one of the instruction sets used during self-supervised training or else was a ‘novel’ formulation. **f**, Performance of partner models in different training regimes given produced instructions or direct input of embedding vectors. Each point represents the average performance of a partner model across tasks using instructions from decoders train with different random initializations. Dots indicate the partner model was trained on all tasks, whereas diamonds indicate performance on held-out tasks. Axes indicate the training regime of the instructing model. Full statistical comparisons of performance can be found in Supplementary Fig. 12.

guide actions in novel settings and the ability to produce a description of a task once it has been learned. RNNs can learn to perform a set of psychophysical tasks simultaneously using a pretrained language transformer to embed a natural language instruction for the current task. Our best-performing models can leverage these embeddings to perform a brand-new model with an average performance of 83% correct. Instructed models that generalize performance do so by leveraging the shared compositional structure of instruction embeddings and task representations, such that an inference about the relations between practiced and novel instructions leads to a good inference about what sensorimotor transformation is required for the unseen task. Finally, we show a network can invert this information and provide a linguistic description for a task based only on the sensorimotor contingency it observes.

Our models make several predictions for what neural representations to expect in brain areas that integrate linguistic information in order to exert control over sensorimotor areas. Firstly, the CCGP analysis of our model hierarchy suggests that when humans must generalize across (or switch between) a set of related tasks based on instructions, the neural geometry observed among sensorimotor mappings should also be present in semantic representations of instructions. This prediction is well grounded in the existing experimental literature where multiple studies have observed the type of abstract structure we find in our sensorimotor-RNNs also exists in sensorimotor areas of biological brains[3,36,37]. Our models theorize that the emergence of an equivalent task-related structure in language areas is essential to instructed action in humans. One intriguing candidate for an area that may support such representations is the language selective subregion of the left inferior frontal gyrus. This area is sensitive to both lexico-semantic and syntactic aspects of sentence comprehension, is implicated in tasks that require semantic control and lies anatomically adjacent to another functional subregion of the left inferior frontal gyrus, which is implicated in flexible cognition[38–41]. We also predict that individual units involved in implementing sensorimotor mappings should modulate their tuning properties on a trial-by-trial basis according to the semantics of the input instructions, and that failure to modulate tuning in the expected way should lead to poor generalization. This prediction may be especially useful to interpret multiunit recordings in humans. Finally, given that grounding linguistic knowledge in the sensorimotor demands of the task set improved performance across models (Fig. 2e), we predict that during learning the highest level of the language processing hierarchy should likewise be shaped by the embodied processes that accompany linguistic inputs, for example, motor planning or affordance evaluation[42].

One notable negative result of our study is the relatively poor generalization performance of GPTNET (XL), which used at least an order of magnitude more parameters than other models. This is particularly striking given that activity in these models is predictive of many behavioral and neural signatures of human language processing[10,11]. Given this, future imaging studies may be guided by the representations in both autoregressive models and our best-performing models to delineate a full gradient of brain areas involved in each stage of instruction following, from low-level next-word prediction to higher-level structured-sentence representations to the sensorimotor control that language informs.

Our models may guide future work comparing compositional representations in nonlinguistic subjects like nonhuman primates. Comparison of task switching (without linguistic instructions) between humans and nonhuman primates indicates that both use abstract rule representations, although humans can make switches much more rapidly[43]. One intriguing parallel in our analyses is the use of compositional rules vectors (Supplementary Fig. 5). Even in the case of nonlinguistic SIMPLENET, using these vectors boosted generalization. Importantly, however, this compositionality is much stronger for our best-performing instructed models. This suggests that language endows agents with a more flexible organization of task subcomponents, which can be recombined in a broader variety of contexts.

Our results also highlight the advantages of linguistic communication. Networks can compress the information they have gained through experience of motor feedback and transfer that knowledge to a partner network via natural language. Although rudimentary in our example, the ability to endogenously produce a description of how to accomplish a task after a period of practice is a hallmark human language skill. The failure to transfer performance by sharing latent representations demonstrates that to communicate information in a group of independent networks of neurons, it needs to pass through a representational medium that is equally interpretable by all members of the group. In humans and for our best-performing instructed models, this medium is language.

A series of works in reinforcement learning has investigated using language and language-like schemes to aid agent performance. Agents receive language information through step-by-step descriptions of action sequences[44,45], or by learning policies conditioned on a language goal[46,47]. These studies often deviate from natural language and receive linguistic inputs that are parsed or simply refer directly to environmental objects. Some larger versions of the pretrained language models we use to embed instructions also display instructions following behavior, that is, GPT-3 (ref. 7), PALM[12], LaMDA[13] and InstructGPT[48] in the modality of language and DALL-E[8] and Stable Diffusion[14] in a language to image modality. The semantic and syntactic understanding displayed in these models is impressive. However, the outputs of these models are difficult to interpret in terms of guiding the dynamics of a downstream action plan. Finally, recent work has sought to engineer instruction following agents that can function in complex or even real-world environments[16–18]. While these models exhibit impressive behavioral repertoires, they rely on perceptual systems that fuse linguistic and visual information making them difficult to compare to language representations in human brains, which emerge from a set of areas specialized for processing language. In all, none of these models offer a testable representational account of how language might be used to induce generalization over sensorimotor mappings in the brain.

Our models by contrast make tractable predictions for what population and single-unit neural representations are required to support compositional generalization and can guide future experimental work examining the interplay of linguistic and sensorimotor skills in humans. By developing interpretable models that can both understand instructions as guiding a particular sensorimotor response, and communicate the results of sensorimotor learning as an intelligible linguistic instruction, we have begun to explain the power of language in encoding and transferring knowledge in networks of neurons.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41593-024-01607-5.

References

1. Cole, M. W. et al. Multi-task connectivity reveals flexible hubs for adaptive task control. *Nature Neurosci.* **16**, 1348–1355 (2013).
2. Miller, E. K. & Cohen, J. D. An integrative theory of prefrontal cortex function. *Annu. Rev. Neurosci.* **24**, 167–202 (2001).
3. Bernardi, S. et al. The geometry of abstraction in the hippocampus and prefrontal cortex. *Cell* **183**, 954–967 (2020).
4. Minxha, J., Adolphs, R., Fusi, S., Mamelak, A. N. & Rutishauser, U. Flexible recruitment of memory-based choice representations by the human medial frontal cortex. *Science* **368**, eaba3313 (2020).

5. Takuya, I. et al. Compositional generalization through abstract representations in human and artificial neural networks. In *Proc. 36th Conference on Neural Information Processing Systems* (eds Koyejo, S. et al.) 32225–32239 (Curran Associates, Inc., 2022).
6. Driscoll, L., Shenoy, K. & Sussillo, D. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. Preprint at *bioRxiv* https://doi.org/10.1101/2022.08.15.503870 (2022).
7. Brown, Tom, et al. Language models are few-shot learners. In *Proc. 34th International Conference on Neural Information Processing Systems* 1877–1901 (Curran Associates Inc., 2020).
8. Ramesh, A. et al. Zero-shot text-to-image generation. In *Proc. 38th International Conference on Machine Learning* (eds Marina, M. & Tong, Z.) 8821–8831 (PMLR, 2021).
9. Radford, A. et al. Language models are unsupervised multitask learners. *OpenAI* **1**, 9 (2019).
10. Schrimpf, M. et al. The neural architecture of language: integrative modeling converges on predictive processing. *Proc. Natl Acad. Sci. USA* https://doi.org/10.1073/pnas.2105646118 (2021).
11. Goldstein, A. et al. Shared computational principles for language processing in humans and deep language models. *Nature Neurosci.* **25**, 369–380 (2022).
12. Chowdhery, A. et al. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.* **24**, 11324–11436 (2023).
13. Thoppilan, R. et al. Lamda: language models for dialog applications. Preprint at https://arxiv.org/abs/2201.08239 (2022).
14. Rombach, R. et al. High-resolution image synthesis with latent diffusion models. In *Proc. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 10674–10685 (IEEE, 2022).
15. Zitkovich, B. et al. Rt-2: vision-language-action models transfer web knowledge to robotic control. In *Proc. 7th Conference on Robot Learning* (eds Tan, J. et al.) 2165-2183 (PMLR, 2023).
16. Abramson, J. et al. Imitating interactive intelligence. Preprint at https://arxiv.org/abs/2012.05672 (2021).
17. DeepMind Interactive Agents Team. Creating multimodal interactive agents with imitation and self-supervised learning. Preprint at https://arxiv.org/abs/2112.03763 (2022).
18. Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T. & Wang, X.-J. Task representations in neural networks trained to perform many cognitive tasks. *Nat. Neurosci.* **22**, 297–306 (2019).
19. Vaswani, A. et al. Attention is all you need. In *Proc. 31st International Conference on Neural Information Processing Systems* 6000–6010 (Curran Associates Inc., 2017).
20. Devlin, J., Chang, M., Lee, K. & Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. Preprint at http://arxiv.org/abs/1810.04805 (2018).
21. Reimers, N. & Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. Preprint at https://arxiv.org/abs/1908.10084 (2019).
22. Bowman, S. R., Angeli, G., Potts, C. & Manning, C. D. A large annotated corpus for learning natural language inference. Preprint at http://arxiv.org/abs/1508.05326 (2015).
23. Radford, A. et al. "Learning transferable visual models from natural language supervision. In *Proc. 38th International Conference on Machine Learning* (eds Marina, M. & Tong, Z.) 8748–8763 (PMLR, 2021).
24. Goel, V., Gold, B., Kapur, S. & Houle, S. Neuroanatomical correlates of human reasoning. *J. Cogn. Neurosci.* **10**, 293–302 (1998).
25. Goel, V., Buchel, C., Frith, C. & Dolan, R. J. Dissociation of mechanisms underlying syllogistic reasoning. *Neuroimage* **12**, 504–514 (2000).
26. Reverberi, C. et al. Neural basis of generation of conclusions in elementary deduction. *Neuroimage* **38**, 752–762 (2007).
27. Noveck, I. A., Goel, V. & Smith, K. W. The neural basis of conditional reasoning with arbitrary content. *Cortex* **40**, 613–622 (2004).
28. Monti, M. M., Osherson, D. N., Martinez, M. J. & Parsons, L. M. Functional neuroanatomy of deductive inference: a language-independent distributed network. *Neuroimage* **37**, 1005–1016 (2007).
29. Monti, M. M., Parsons, L. M. & Osherson, D. N. The boundaries of language and thought in deductive inference. *Proc. Natl Acad. Sci. USA* **106**, 12554–12559 (2009).
30. Coetzee, J. P. & Monti, M. M. At the core of reasoning: dissociating deductive and non-deductive load. *Hum. Brain Mapp.* **39**, 1850–1861 (2018).
31. Monti, M. M. & Osherson, D. N. Logic, language and the brain. *Brain Res.* **1428**, 33–42 (2012).
32. Prado, J. The relationship between deductive reasoning and the syntax of language in broca's area: a review of the neuroimaging literature. *L'année Psychol.* **118**, 289–315 (2018).
33. Ito, T., Yang, G. R., Laurent, P., Schultz, D. H. & Cole, M. W. Constructing neural network models from brain data reveals representational transformations linked to adaptive behavior. *Nat. Commun.* **13**, 673 (2022).
34. Shadlen, M. N. & Newsome, W. T. Neural basis of a perceptual decision in the parietal cortex (area lip) of the rhesus monkey. *J. Neurophysiol.* **86**, 1916–1936 (2001).
35. Huk, A. C. & Shadlen, M. N. Neural activity in macaque parietal cortex reflects temporal integration of visual motion signals during perceptual decision making. *J. Neurosci.* **25**, 10420–10436 (2005).
36. Panichello, M. F. & Buschman, T. J. Shared mechanisms underlie the control of working memory and attention. *Nature* **592**, 601–605 (2021).
37. Nieh, E. H. et al. Geometry of abstract learned knowledge in the hippocampus. *Nature* **595**, 80–84 (2021).
38. Fedorenko, E. & Blank, I. A. Broca's area is not a natural kind. *Trends Cogn. Sci.* **24**, 270–284 (2020).
39. Fedorenko, E., Duncan, J. & Kanwisher, N. Language-selective and domain-general regions lie side by side within broca's area. *Curr. Biol.* **22**, 2059–2062 (2012).
40. Gao, Z. et al. Distinct and common neural coding of semantic and non-semantic control demands. *NeuroImage* **236**, 118230 (2021).
41. Duncan, J. The multiple-demand (MD) system of the primate brain: mental programs for intelligent behaviour. *Trends Cogn. Sci.* **14**, 172–179 (2010).
42. Buccino, G., Colagé, I., Gobbi, N. & Bonaccorso, G. Grounding meaning in experience: a broad perspective on embodied language. *Neurosci. Biobehav. Rev.* **69**, 69–78 (2016).
43. Mansouri, F. A., Freedman, D. J. & Buckley, M. J. Emergence of abstract rules in the primate brain. *Nat. Rev. Neurosci.* **21**, 595–610 (2020).
44. Oh, J. Singh, S., Lee, H. & Kohli, P. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proc. 34th International Conference on Machine Learning* 2661–2670 (JMLR.org, 2017).
45. Chaplot, D. S., Mysore Sathyendra, K., Pasumarthi, R. K., Rajagopal, D., & Salakhutdinov, R. Gated-attention architectures for task-oriented language grounding. In *Proc. 32nd AAAI Conference on Artificial Intelligence* Vol. 32 (AAAI Press, 2018).
46. Sharma, P., Torralba, A. & Andreas, J. Skill induction and planning with latent language. Preprint at https://arxiv.org/abs/2110.01517(2021).
47. Jiang, Y., Gu, S., Murphy, K. & Finn, C. Language as an abstraction for hierarchical deep reinforcement learning. In *Proc. 33rd International Conference on Neural Information Processing Systems* 9419–943132 (Curran Associates Inc., 2019).
48. Ouyang, L. et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems* 27730–27744 (Curran Associates, Inc., 2022).

Methods

Model architecture

Sensorimotor-RNN. The base model architecture and task structure used in this paper follows[18]. All networks of sensorimotor units denoted sensorimotor-RNN are gated recurrent units (GRU)[49] using rectified linear unit (ReLU) nonlinearities with 256 hidden units each. Inputs to the networks consist of (1) sensory inputs, X_t and (2) task-identifying information, I_t. We initialize hidden activity in the GRU as $h^0 \in \mathbb{R}^{256}$ with values set to 0.1. All networks of sensorimotor units use the same hidden state initialization, so we omit h^0 in network equations. At each time step, a readout layer $\text{Linear}_{\text{out}}$ decodes motor activity, $\hat{y}_t$, from the activity of recurrent hidden units, h_t, according to:

$$h_t = \text{SensorimotorRNN}\left(X_t, I_t; h_{t-1}\right) \qquad h_t \in \mathbb{R}^{256}$$

$$\hat{y}_t = \sigma\left(\text{Linear}_{\text{out}}(h_t)\right) \qquad \hat{y}_t \in \mathbb{R}^{33}$$

where σ denotes the sigmoid function. Sensory inputs X_t are made up of three channels, two sensory modalities $x_{\text{mod}\,1,t}$ and $x_{\text{mod}\,2,t}$, and a fixation channel $x_{\text{fix},t}$. Both $x_{\text{mod}\,1,t}, x_{\text{mod}\,2,t} \in \mathbb{R}^{32}$ and stimuli in these modalities are represented as hills of activity with peaks determined by units' preferred directions around a one-dimensional circular variable. For an input at direction θ, the activity of a given input unit u_i with preferred direction θ_i is

$$u_i = str \times 0.8 \exp\left[-0.5 \times \left(\frac{8|\theta - \theta_i|}{\pi}\right)^2\right]$$

where str is the coefficient describing stimulus strength. The fixation channel $x_{\text{fix},t} \in \mathbb{R}^1$ is a single unit simulating a fixation cue for the network. In all, sensory input $X_t = (x_{mod1,t}, x_{mod2,t}, x_{fix,t}) \in \mathbb{R}^{65}$. Motor output, $\hat{y}_t$ consists of both a 32-dimensional ring representing directional responses to the input stimulus as well as a single unit representing model fixation, so that $\hat{y}_t \in \mathbb{R}^{33}$.

For all models, task-identifying information $I_t \in \mathbb{R}^{64}$. Task-identifying information is presented throughout the duration of a trial and remains constant such that $I_t = I_{t'} \forall t, t'$. For all models, task-identifying info I_t and sensory input X_t are concatenated as inputs to the sensorimotor-RNN.

Nonlinguistic models. For SIMPLENET, we generate a set of 64-dimensional orthogonal task rules by constructing an orthogonal matrix using the Python package scipy.stats.ortho_group, and assign rows of this matrix to each task type. For STRUCTURENET, we generate a set of ten orthogonal, 64-dimensional vectors in the same manner, and each of these represents a dimension of the task set (that is, respond weakest versus strongest direction, respond in the same versus opposite direction, pay attention only to stimuli in the first modality, and so on). Rule vectors for tasks are then simple combinations of each of these ten basis vectors. For a full description of structure rule vectors, see Supplementary Note 3.

We also test SIMPLENETPLUS and STRUCTURENETPLUS, which use an additional hidden layer with 128 units and ReLU nonlinearities to process orthogonal tasks rules I_t into a vector $\bar{I}_t$ which is used by sensorimotor-RNN as task-identifying information.

$$\bar{I}_t' = \text{ReLU}(\text{Linear}_{\text{RuleEmb1}}(\text{I}_t)) \quad \bar{I}_t' \in \mathbb{R}^{128}$$

$$\bar{I}_t' = \text{ReLU}(\text{Linear}_{\text{RuleEmb2}}(\text{I}_t')) \quad \bar{I}_t' \in \mathbb{R}^{128}$$

$$\bar{I}_t = \text{ReLU}(\text{Linear}_{\text{RuleEmb3}}(\text{I}_t{}')) \quad \bar{I}_t \in \mathbb{R}^{64}$$

Full results for these models are included in Supplementary Fig. 4.

Pretrained transformers. The main language models we test use pretrained transformer architectures to produce I. Importantly, transformers differ in the type of pretraining objective used to tune the model parameters. GPT is trained to predict the next word given a context of words[9]. GPT (XL) follows the same objective but trains for longer on a larger dataset[50]. Both models are fully autoregressive. BERT, by contrast, takes bidirectional language inputs and is tasked with predicting masked words that appear in the middle of input phrases. Additionally, BERT is trained on a simple sentence prediction task where the model must determine if input sentence 1 is followed by input sentence 2 in the training corpus. Extending this principle, SBERT is explicitly trained to produce fixed-length embeddings of whole sentences[21]. It takes pretrained BERT networks and uses them in a siamese architecture[51], which allows the weights of the model to be tuned in a supervised fashion according to the Stanford Natural Language Inference dataset[22]. Natural language inference is a three-way categorization task where the network must infer the logical relationship between sentences: whether a premise sentence implies, contradicts or is unrelated to a hypothesis sentence. Finally, CLIP is trained to jointly embed images and language[23]. It uses data from captioned images and is asked to properly categorize which text and images pairs match or are mismatched in the dataset via a contrastive loss.

Importantly, the natural output of a transformer is a matrix of size $\text{dim}_{\text{trans.}} \times \mathcal{T}$, the inherent dimensionality of the transformer by the length of the input sequence. To create an embedding space for sentences it is standard practice to apply a pooling method to the transformer output, which produces a fixed-length representation for each instruction.

For GPT, GPT (XL), BERT and SBERT, we use an average pooling method. Suppose we have an input instruction $w_1 \ldots w_{\mathcal{T}}$. Following standard practice with pretrained language models, the input to our transformers is tokenized with special 'cls' and 'eos' tokens at the beginning and end of the input sequence. We then compute I as follows:

$$h^{\text{tran.}} = \text{transformer}\left([\text{cls}], w_1 \ldots w_{\mathcal{T}}, [\text{eos}]\right), \qquad h^{\text{tran.}} \in \mathbb{R}^{\text{dim}_{\text{trans.}} \times \mathcal{T}+2}$$

$$h^I = \text{mean}(h^{\text{tran.}}), \qquad h^I \in \mathbb{R}^{\text{dim}_{\text{trans.}}}$$

$$I = \text{Linear}_{\text{embed}}(h^I) \qquad I \in \mathbb{R}^{64}$$

We chose this average pooling method primarily because a previous study[21] found that this resulted in the highest-performing SBERT embeddings. Another alternative would be to simply use the final hidden representation of the 'cls' token as a summary of the information in the entire sequence (given that BERT architectures are bidirectional, this token will have access to the whole sequence).

$$h^{\text{tran.}} = \text{transformer}\left([\text{cls}], w_1 \ldots w_{\mathcal{T}}, [\text{eos}]\right), \qquad h^{\text{tran.}} \in \mathbb{R}^{\text{dim}_{\text{trans.}} \times \mathcal{T}+2}$$

$$h^I = (h^{\text{tran.}}_{\text{cls}}) \qquad h^I \in \mathbb{R}^{\text{dim}_{\text{trans.}}}$$

Where $h^{\text{tran.}}_{\text{cls}}$ denote the last hidden representation for the 'cls' token. Ref. 21 found this pooling method performed worse than average pooling, so we don't include these alternatives in our results. For GPT and GPT (XL), we also tested a pooling method where the fixed-length representation for a sequence was taken from the transformer output of the 'eos' token. In this case:

$$h^{\text{tran.}} = \text{transformer}\left([\text{cls}], w_1 \ldots w_{\mathcal{T}}, [\text{eos}]\right), \qquad h^{\text{tran.}} \in \mathbb{R}^{\text{dim}_{\text{trans.}} \times \mathcal{T}+2}$$

$$h^I = (h^{\text{tran.}}_{\text{eos}}), \qquad h^I \in \mathbb{R}^{\text{dim}_{\text{trans.}}}$$

$$I = \text{Linear}_{\text{embed}}(h^I), \qquad I \in \mathbb{R}^{64}$$

We found that GPT failed to achieve even a relaxed performance criterion of 85% across tasks using this pooling method, and GPT (XL) performed worse than with average pooling, so we omitted these models from the main results (Supplementary Fig. 11). For CLIP models we use the same pooling method as in the original multiModal training procedure, which takes the outputs of the [cls] token as described above.

For all the above models, we also tested a version where the information from the pretrained transformers is passed through a multilayer perceptron with a single hidden layer of 256 hidden units and ReLU nonlinearities. We found that this manipulation reduced performance across all models, verifying that a simple linear embedding is beneficial to generalization performance.

For GPT, BERT and SBERT, $\mathrm{dim}_{\mathrm{trans.}} = 768$ and each model uses a total of ~100 million parameters; for SBERT (L) $\mathrm{dim}_{\mathrm{trans.}} = 1,024$ and the model uses ~300 million parameters; GPT (XL) $\mathrm{dim}_{\mathrm{trans.}} = 1,600$ and the model uses ~1.5 billion parameters; for CLIP, $\mathrm{dim}_{\mathrm{trans.}} = 512$ and the model uses ~60 million parameters. Full PyTorch implementations, including all pretrained weights and model hyperparameters, can be accessed at the Huggingface library (https://huggingface.co/docs/transformers/)[52].

BoW model. For our BoW model, instructions are represented as a vector of binary activations the size of the instruction vocabulary, where each unit indicates the inclusion or exclusion of the associated word in the current instruction. For our instruction set, |vocab| = 181. This vector is then projected through a linear layer into 64-dimensional space.

$$h_i^{\mathrm{BoW}} = \begin{cases} 1 & \text{if } w_i \in (w_1 \dots w_{\mathcal{T}}) \\ 0 & \text{otherwise} \end{cases} \qquad h^{\mathrm{BoW}} \in \mathbb{R}^{|\mathrm{vocab}|}$$

$$I = \mathrm{Linear}_{\mathrm{embed}}(h^{\mathrm{BoW}}), \qquad I \in \mathbb{R}^{64}$$

Blank slate language models. Given that tuning the last layers of language models resulted in improved performance (Fig. 2e), we tested two additional models to determine if training a blank slate language model trained exclusively on the loss from sensorimotor tasks would improve performance. These models consist of passing BoW representations through a multilayer perceptron and passing pretrained BERT word embeddings through one layer of a randomly initialized BERT encoder. Both models performed poorly compared to pretrained models (Supplementary Fig. 4.5), confirming that language pretraining is essential to generalization.

Tasks sets

Tasks were divided into five interrelated subgroups: 'go', 'decision-making', 'matching', and 'comparison' and 'duration'. Depending on the task, multiple stimuli may appear during the stimulus epoch. Also, depending on the task, models may be required to respond in a particular direction or repress response altogether. Unless otherwise specified, zero-mean Gaussian noise is added independently at each time step and to each input unit and the variance of this noise is drawn randomly from $\mathbb{U}[0.1, 0.15]$. The timing of stimuli differs among the tasks type. However, for all tasks, trials can be divided into preparatory, stimulus and response epochs. The stimulus epoch can be subdivided into three parts—stim1, delay and stim23—although these distinct parts aren't used by all tasks. A trial lasts for a total of $T = 150$ time steps. Let dur_{epoch} denote the duration in simulated time steps of a given epoch. Then

$$dur_{\mathrm{response}} \sim \left\{ i | 20 < i \le 25; i \in \mathbb{N} \right\}$$

$$dur_{\mathrm{stim1}}, dur_{\mathrm{stim2}} \sim \left\{ i | 37 < i \le 50; i \in \mathbb{N} \right\}$$

$$dur_{\mathrm{delay}} \sim \left\{ i | 15 < i \le 25; i \in \mathbb{N} \right\}$$

$$dur_{\mathrm{prep.}} = 150 - \left(dur_{\mathrm{response}} + dur_{\mathrm{stim1}} + dur_{\mathrm{stim2}} + dur_{\mathrm{delay}} \right)$$

For tasks that don't utilize a delay structure, stim1, stim2 and delay epochs are grouped together in a single stimulus epoch where $dur_{\mathrm{stimulus}} = dur_{\mathrm{stim1}} + dur_{\mathrm{stim2}} + dur_{\mathrm{delay}}$. Unless otherwise specified, a fixation cue with a constant strength $str_{\mathrm{fix}} = 1$ is activated throughout the preparatory and stimulus epochs. For example trials of each task, see Supplementary Fig. 13.

'Go' tasks. The 'Go' family of tasks includes 'Go', 'RTGo', 'AntiGo', 'AntiRTGo' and modality-specific versions of each task denoted with either 'Mod1' and 'Mod2'. In both the 'Go' and 'AntiGo' tasks, a single stimulus is presented at the beginning of the stimulus epoch. The direction of the presented stimulus is generated by drawing from a uniform distribution between 0 and 2π, that is, $\theta_{\mathrm{stim}} \sim \mathbb{U}[0, 2\pi]$. The stimulus will appear in either modality 1 or modality 2 with equal probability. The strength of the stimulus is given by $str_{\mathrm{stim}} \sim \mathbb{U}[1.0, 1.2]$. In the 'Go' task, the target response is in the same direction as the presented stimulus, that is, $\theta_{\mathrm{stim}} = \theta_{\mathrm{target}}$, while in the 'AntiGo' task the direction of the response should be in the opposite of the stimulus direction, $\theta_{\mathrm{stim}} + \pi = \theta_{\mathrm{target}}$. For modality-specific versions of each task, a stimulus direction is drawn in each modality $\theta_{\mathrm{stim,mod1}} \sim \mathbb{U}[0, 2\pi]$ and $\theta_{\mathrm{stim,mod2}} \sim \mathbb{U}[0, 2\pi]$ and for modality-specific Go-type tasks

$$\theta_{\mathrm{target}} = \begin{cases} \theta_{\mathrm{stim,mod1}} & \text{if Mod1 task} \\ \theta_{\mathrm{stim,mod2}} & \text{if Mod2 task} \end{cases}$$

while for modality-specific AntiGo-type tasks

$$\theta_{\mathrm{target}} = \begin{cases} \theta_{\mathrm{stim,mod1}} + \pi & \text{if Mod1 task} \\ \theta_{\mathrm{stim,mod2}} + \pi & \text{if Mod2 task} \end{cases}$$

For 'RT' versions of the 'Go' tasks, stimuli are only presented during the response epoch and the fixation cue is never extinguished. Thus, the presence of the stimulus itself serves as the response cue and the model must respond as quickly as possible. Otherwise, stimuli persist through the duration of the stimulus epoch.

'Decision-making' tasks. The 'decision-making' family of tasks includes 'DM' (decision-making), 'AntiDM', 'MultiDM' (multisensory decision-making), 'AntiMultiDM,' modality-specific versions of each of these tasks and, finally, confidence-based versions of 'DM' and 'AntiDM.' For all tasks in this group, two stimuli are presented simultaneously and persist throughout the duration of the stimulus epoch. They are drawn according to $\theta_{\mathrm{stim1}} \sim \mathbb{U}[0, 2\pi]$ and $\theta_{\mathrm{stim2}} \sim \mathbb{U}[(\theta_{\mathrm{stim1}} - 0.2\pi, \theta_{\mathrm{stim1}} - 0.6\pi) \cup (\theta_{\mathrm{stim1}} + 0.2\pi, \theta_{\mathrm{stim1}} + 0.6\pi)]$ A base strength applied to both stimuli is drawn such that $str_{\mathrm{base}} \sim \mathbb{U}[1.0, 1.2]$. A contrast is drawn from a discrete distribution such that $c \sim \{-0.175, -0.15, -0.1, 0.1, 0.15, 0.175\}$ so the stimulus strength associated with each direction in a trial are given by $str_{\mathrm{stim1}} = str_{\mathrm{base}} + c$ and $str_{\mathrm{stim2}} = str_{\mathrm{base}} - c$.

For the 'DM' task,

$$\theta_{\mathrm{target}} = \begin{cases} \theta_{\mathrm{stim1}} & \text{if } str_{\mathrm{stim1}} > str_{\mathrm{stim2}} \\ \theta_{\mathrm{stim2}} & \text{otherwise} \end{cases}$$

and for the the 'AntiDM' task,

$$\theta_{\mathrm{target}} = \begin{cases} \theta_{\mathrm{stim1}} & \text{if } str_{\mathrm{stim1}} < str_{\mathrm{stim2}} \\ \theta_{\mathrm{stim2}} & \text{otherwise} \end{cases}$$

For these versions of the tasks, the stimuli are presented in either modality 1 or modality 2 with equal probability. For the multisensory versions of each task, stimuli directions are drawn in the same manner and presented across both modalities so that $\theta_{\mathrm{stim1,mod1}} = \theta_{\mathrm{stim1,mod2}}$ and $\theta_{\mathrm{stim2,mod1}} = \theta_{\mathrm{stim2,mod2}}$. Base strengths are drawn independently for each modality. Contrasts for both modalities are drawn from a discrete distribution such that $c_{\mathrm{mod\,1}}, c_{\mathrm{mod\,2}} \sim \{0.2, 0.175, 0.15, 0.125, -0.125, -0.15, -0.175, -0.2\}$. If both $|c_{\mathrm{mod\,1}}| - |c_{\mathrm{mod\,2}}| = 0$ then contrasts are redrawn to avoid zero-contrast trials during training. If both $c_{\mathrm{mod\,1}}$ and $c_{\mathrm{mod\,2}}$ have the same sign, then contrasts are redrawn to ensure that the trial requires integrating over both modalities as opposed to simply performing a 'DM' task in a single modality. Criteria for target responses are measured as the strength of a given direction summed over both modalities. So, for 'MultiDM'

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim1,mod1}} & \text{if } str_{\text{stim1,mod1}} + str_{\text{stim1,mod2}} > str_{\text{stim2,mod1}} \\ & +str_{\text{stim2,mod2}} \\ \theta_{\text{stim2,mod1}} & \text{otherwise} \end{cases}$$

and for 'AntiMultiDM'

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim1,mod1}} & \text{if } str_{\text{stim1,mod1}} + str_{\text{stim1,mod2}} < str_{\text{stim2,mod1}} \\ & +str_{\text{stim2,mod2}} \\ \theta_{\text{stim2,mod1}} & \text{otherwise} \end{cases}$$

Stimuli for modality-specific versions of each task are generated in the same way as multisensory versions of the task. Criteria for target response are the same as standard versions of 'DM' and 'AntiDM' tasks applied only to stimuli in the relevant modality.

In confidence-based decision-making tasks ('ConDM' and 'ConAntiDM'), the stimuli directions are drawn in the same way as above. Stimuli are shown in either modality 1 or modality 2 with equal probability. In each trial, $str_{\text{base}} = 1$. The contrast and noise for each trial is based on the thresholded performance of a SIMPLENET model trained on all tasks except 'ConDM' and 'ConAntiDM'. Once this model has been trained, we establish a threshold across levels of noise and contrasts for which the model can perform a 'DM' or an 'AntiDM' task at 95% correct. We then draw contrasts and noises for trials from above and below this threshold with equal probability during training. In trials where the noise and contrast levels fell below the 95% correct threshold, the model must repress response, and otherwise perform the decision-making task (either 'DM' or 'AntiDM').

'Comparison' tasks. Our comparison task group includes 'COMP1', 'COMP2', 'MultiCOMP1', 'MultiCOMP2', 'Anti' versions of each of these tasks, as well as modality-specific versions of 'COMP1' and 'COMP2' tasks. This group of tasks is designed to extend the basic decision-making framework into a setting with more complex control demands. These tasks utilize the delay structure in the stimulus epoch so that stim1 appears only during the stim1 epoch, followed by a delay, and finally stim2. This provides a temporal ordering on the stimuli. In 'COMP1', the model must respond to the first stimulus only if it has greater strength than the second and otherwise repress a response that is

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim1}} & \text{if } str_{\text{stim1}} > str_{\text{stim2}} \\ \text{repress} & \text{otherwise} \end{cases}$$

Likewise, in 'COMP2', the model must respond to the second direction if it presented with greater strength than the first otherwise repress response that is

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim2}} & \text{if } str_{\text{stim2}} > \text{str}_{\text{stim1}} \\ \text{repress} & \text{otherwise} \end{cases}$$

In 'Anti' versions of the task the ordering criteria is the same except for stimuli with least strength, that is, for 'AntiCOMP1'

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim1}} & \text{if } \text{str}_{\text{stim1}} < \text{str}_{\text{stim2}} \\ \text{repress} & \text{otherwise} \end{cases}$$

and for 'AntiCOMP2'

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim2}} & \text{if } \text{str}_{\text{stim2}} < \text{str}_{\text{stim1}} \\ \text{repress} & \text{otherwise} \end{cases}$$

In multisensory settings, the criteria for target direction are analogous to the multisensory decision-making tasks where strength is integrated across modalities. Likewise, for modality-specific versions, the criteria are only applied to stimuli in the relevant modality. Stimuli directions and strength for each of these tasks are drawn from the same distributions as the analogous task in the 'decision-making' family. However, during training, we make sure to balance trials where responses are required and trials where models must repress response.

'Duration' tasks. The 'duration' family of tasks includes 'Dur1', 'Dur2', 'MultiDur1', 'MultiDur2', 'Anti' versions of each of these tasks and modality-specific versions of 'Dur1' and 'Dur2' tasks. These tasks require models to perform a time estimation task with the added demand or stimuli ordering determining relevance for response. Like in 'comparison' tasks, stim1 is presented followed by a delay and then stim2. For 'Dur1' trials

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim1}} & \text{if } dur_{\text{stim1}} > dur_{\text{stim2}} \\ \text{repress} & \text{otherwise} \end{cases}$$

Likewise, for 'Dur2'

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim2}} & \text{if } dur_{\text{stim2}} > dur_{\text{stim1}} \\ \text{repress} & \text{otherwise} \end{cases}$$

In 'Anti' versions of these tasks, the correct response is in the direction of the stimulus with the shortest duration given the ordering criteria is met. Hence, for 'AntiDur1'

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim1}} & \text{if } dur_{\text{stim1}} < dur_{\text{stim2}} \\ \text{repress} & \text{otherwise} \end{cases}$$

and for 'AntiDur2'

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim2}} & \text{if } dur_{\text{stim2}} < dur_{\text{stim1}} \\ \text{repress} & \text{otherwise} \end{cases}$$

Across these tasks directions are drawn according to $\theta_{\text{stim1}} \sim \mathbb{U}[0, 2\pi]$ and $\theta_{\text{stim2}} \sim \mathbb{U}[(\theta_{\text{stim1}} - 0.2\pi, \theta_{\text{stim1}} - 0.6\pi) \cup (\theta_{\text{stim1}} + 0.2\pi, \theta_{\text{stim1}} + 0.6\pi)]$. Stimulus strengths are drawn according to $str_{\text{stim1}}, str_{\text{stim2}} \sim \mathbb{U}[0.8, 1.2]$. To set the duration of each stimulus, we first draw $dur_{\text{long}} \sim \{i|35 < i \leq 50, i \in \mathbb{N}\}$ and $dur_{\text{short}} \sim \{i|25 < i \leq (dur_{\text{long}} - 8), i \in \mathbb{N}\}$. During training, we determine which trials for a given task should and should not require a response in order to evenly balance repress and respond trials. We then assign dur_{long} and dur_{short} to either stim1 or stim2 so that the trial requires the appropriate response given the particular task type.

Again, criteria for correct response in the multisensory and modality-specific versions of each tasks follow analogous tasks in the 'decision-making' and 'comparison' groups where multisensory versions of the task require integrating total duration over each modality, and modality-specific tasks require only considering durations in the given task modality. For multisensory tasks, we draw duration value $dur_{\text{long}} \sim \{i|75 < i \leq 100, i \in \mathbb{N}\}$ and then split this value $dur_{\text{long0}} = dur_{\text{long}} \times 0.55$ and $dur_{\text{long1}} = dur_{\text{long}} \times 0.45$. We also draw a value $dur_{\text{short}} = dur_{\text{long}} - \Delta dur$ where $\Delta dur \sim \{i|15 < i \leq 25, i \in \mathbb{N}\}$. This value is then subdivided further into $dur_{\text{short0}} = dur_{\text{long1}} + \Delta dur_{\text{short}}$ where $\Delta dur_{\text{short}} \sim \{i|19 < i \leq 15, i \in \mathbb{N}\}$ and $dur_{\text{short1}} = dur_{\text{Short}} - dur_{\text{short0}}$. Short and long durations can then be allocated to the ordered stimuli according to task type. Drawing durations in this manner ensures that, like in 'decision-making' and 'comparison' groups, correct answers truly require models to integrate durations over both modalities, rather than simply performing the task in a given modality to achieve correct responses.

'Matching' tasks. The 'matching' family of tasks consists of 'DMS' (delay match to stimulus), 'DNMS' (delay non-match to stimulus), 'DMC' (delay match to category) and 'DMNC' (delay non-match to category) tasks. For all tasks, stim1 is presented at the beginning of the stimulus epoch, followed by a delay, and the presentation of stim2. The stimulus strength is drawn according to $str_{\text{stim1}}, str_{\text{stim2}} \sim \mathbb{U}[0.8, 1.2]$. The input modality for any given trial is chosen at random with equal probability. In both 'DMS' and 'DNMS' tasks, trials are constructed as 'matching stim' trials or 'mismatching stim' trials with equal probability. In 'matching stim' trials $\theta_{\text{stim1}} \sim \mathbb{U}[0, 2\pi]$ and $\theta_{\text{stim2}} = \theta_{\text{stim1}}$. In 'mismatch stim' trials, $\theta_{\text{stim1}} \sim \mathbb{U}[0, 2\pi]$ and

$$\theta_{\text{stim2}} \sim \mathbb{U}[(\theta_{\text{stim1}} - 0.2\pi, \theta_{\text{stim1}} - 0.6\pi) \cup (\theta_{\text{stim1}} + 0.2\pi, \theta_{\text{stim1}} + 0.6\pi)].$$

For 'DMS', models must respond in the displayed direction if the stimuli match, otherwise repress response,

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim1}} & \text{if } \theta_{\text{stim1}} = \theta_{\text{stim2}} \\ \text{repress} & \text{otherwise} \end{cases}$$

and for 'DNMS', models must respond to the second direction if both directions are mismatched,

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim2}} & \text{if } \theta_{\text{stim1}} \neq \theta_{\text{stim2}} \\ \text{repress} & \text{otherwise} \end{cases}$$

'DMC' and 'DNMC' tasks are organized in a similar manner. The stimulus input space is divided evenly into two categories such that cat1 = $\{\theta: 0 < \theta \leq \pi\}$ and cat2 = $\{\theta: \pi < \theta \leq 2\pi\}$. For 'DMC' and 'DNMC' tasks, trials are constructed as 'matching cat.' trials or 'mismatching cat.' trials with equal probability. In 'matching cat.' trials $\theta_{\text{stim1}} \sim \mathbb{U}[0, 2\pi]$ and $\theta_{\text{stim2}} \sim \mathbb{U}(\text{cat}_{\text{stim1}})$, where $\mathbb{U}(\text{cat}_{\text{stim1}})$ is a uniform draw from the category of stim1. In 'mismatch stim' trials, $\theta_{\text{stim1}} \sim \mathbb{U}[0, 2\pi]$ and $\theta_{\text{stim2}} \sim \mathbb{U}(-\text{cat}_{\text{stim1}})$ where $-\text{cat}_{\text{stim1}}$ is the opposite category as stim1. For 'DMC', the model must respond in the first direction if both stimuli are presented in the same category otherwise repress response,

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim1}} & \text{if } \text{cat}_{\text{stim1}} = \text{cat}_{\text{stim2}} \\ \text{repress} & \text{otherwise} \end{cases}$$

and for 'DNMC', the model should respond to the second direction if both stimuli are presented in opposite categories otherwise repress response,

$$\theta_{\text{target}} = \begin{cases} \theta_{\text{stim2}} & \text{if } \text{cat}_{\text{stim1}} \neq \text{cat}_{\text{stim2}} \\ \text{repress} & \text{otherwise} \end{cases}$$

Target output and correct criteria

The target output $y \in \mathbb{R}^{33 \times T}$ for a trial entails maintaining fixation in $y_1 = y_{\text{fix}}$ during the stimulus epoch, and then either responding in the correct direction or repressing activity in the remaining target response units $y_{2...33}$ in the response epoch. Since the model should maintain fixation until response, target for fixation is set at $y_{\text{fix}} = 0.85$ during preparatory and stimulus epochs and $y_{\text{fix}} = 0.05$ in the response epoch. When a response is not required, as in the preparatory and stimulus epochs and with repressed activity in the response epoch, unit i takes on a target activity of $y_i = 0.05$. Alternatively, when there is a target direction for response,

$$y_i = 0.8 \exp\left[-0.5 \times \left(\frac{8|\theta_{\text{target}} - \theta_i|}{\pi}\right)^2\right] + 0.05$$

where θ_i is the preferred direction for unit i. Like in sensory stimuli, preferred directions for target units are evenly spaced values from $[0, 2\pi]$ allocated to the 32 response units.

For a model response to count as correct, it must maintain fixation, that is, $\hat{y}_{\text{fix}} > 0.5$ during preparatory and stimulus epochs. When no response is required $\hat{y}_i < 0.15$. When a response is required, response activity is decoded using a population vector method and $\theta_{\text{resp.}} \in (\theta_{\text{target}} - \frac{\pi}{10}, \theta_{\text{target}} + \frac{\pi}{10})$. If the model fails to meet any of these criteria, the trial response is incorrect.

Model training

Again following ref. 18, model parameters are updated in a supervised fashion according to a masked mean squared error loss (mMSE) computed between the model motor response, $\hat{y}_{1...T} = \hat{y}$, and the target, $y_{1...T} = y$, for each trial.

$$L = \text{mMSE}(y, \hat{y}) = \text{mask} \times \left\langle (y_t - \hat{y}_t)^2 \right\rangle_t$$

Here, the multiplication sign denotes element-wise multiplication. Masks weigh the importance of different trial epochs. During preparatory and stimulus epochs, mask weights are set to 1; during the first five time steps of the response epoch, the mask value is set to 0; and during the remainder of the response epoch, the mask weight is set to 5. The mask value for the fixation is twice that of other values at all time steps.

For all models, we update Θ = {sensorimotor-RNN, Linear$_{\text{out}}$} during training on our task set. For instructed models, we additionally update Linear$_{\text{embed}}$ in the process of normal training. We train models using standard PyTorch machinery and an Adam optimizer. An epoch consists of 2,400 mini-batches, with each mini-batch consisting of 64 trials. For all models, we use the same initial learning rate as in ref. 18, $lr = 0.001$. We found that in the later phases of training, model performance oscillated based on which latest task presented during training, so we decayed the learning rate for each epoch by a factor of $\gamma = 0.95$, which allowed performance to converge smoothly. Following ref. 18, models train until they reach a threshold performance of 95% across all tasks (and train for a minimum of 35 epochs). We found that training for GPTNET tended to asymptote below performance threshold for multisensory versions of comparison tasks. This held true over a variety of training hyperparameters and learning rate scheduler regimes. Hence, we relax the performance threshold of GPTNET to 85%. For each model type, we train five models that start from five different random initializations. Where applicable, results are averaged over these initializations.

Language model fine-tuning. When fine-tuning models, we allow the gradient from the motor loss experienced during sensorimotor training to fine-tune the weights in the final layers of the transformer language models. During normal training, we checkpoint a copy of our instructed models after training for 30 epochs. We then add the last three transformer layers to the set of trainable parameters, and reset the learning rates to $lr = 1 \times 10^{-4}$ for Θ = {sensorimotor-RNN, Linear$_{\text{out}}$} and $lr^{\text{lang}} = 3 \times 10^{-4}$ for Θ^{lang} = {Linear$_{\text{embed}}$, transformer$_{-3,-2,-1}$} where transformer$_{-3,-2,-1}$ denotes the parameters of the last three layers of the relevant transformer architecture. We used these reduced learning rates to avoid completely erasing preexisting linguistic knowledge. Similarly for RNN parameters, we found the above learning rate avoided catastrophic forgetting of sensorimotor knowledge while also allowing the RNN to adapt to updated language embeddings across all models. Autoregressive models were much more sensitive to this procedure, often collapsing at the beginning of fine-tuning. Hence, for GPTNETXL and GPTNET, we used $lr^{\text{lang}} = 5 \times 10^{-5}$, which resulted in robust learning. Models train until they reach a threshold performance of 95% across training tasks or 85% correct for GPTNET.

Hold-out testing

During hold-out testing, we present models with 100 batches of one of the tasks that had been held out of training. For the instructed model, the only weights allowed to update during this phase are Θ = {sensorimotor-RNN, Linear$_{\text{out}}$, Linear$_{\text{embed}}$}. All weights of SIMPLENET and STRUCTURENET are trainable in this context. In this hold-out setting, we found that in more difficult tasks for some of our more poorly performing models, the standard hyperparameters we used during training resulted in unstable learning curves for novel tasks. To stabilize performance and thereby create fair comparisons across models, we used an increased batch size of 256. We then began with the standard learning rate of 0.001 and decreased this by increments of 0.0005 until all models showed robust learning curves. This resulted in a learning rate of 8×10^{-4}. All additional results shown in the Supplementary Information section 4 follow this procedure.

CCGP calculation

To calculate CCGP, we trained a linear decoder on a pair of tasks and then tested that decoder on alternative pairs of tasks that have an analogous relationship. We grouped tasks into eight dichotomies: 'Go' versus 'Anti', 'Standard' versus 'RT', 'Weakest' versus 'Strongest', 'Longest' versus 'Shortest', 'First Stim.' versus 'Second Stim', 'Stim Match' versus 'Category Match', 'Matching' versus 'Non-Matching' and 'Mod1' versus 'Mod2'. As an example, the 'Go' versus 'Anti' dichotomy includes ('Go', 'AntiGo'), ('GoMod1', 'AntiGoMod1'), ('GoMod2', 'AntiGoMod2'), ('RTGo', 'AntiRTGo'), ('RTGoMod1', 'AntiRTGoMod1') and ('RTGoMod2', 'AntiRTGoMod2') task pairs. For 'RNN' task representations, we extracted activity at the time of stimulus onset for 250 example trials. For language representations, we input the instruction sets for relevant tasks to our language model and directly analyze activity in the 'embedding' layer or take the sequence-averaged activity in each transformer layer. For nonlinguistic models, we simply analyze the space of rule vectors. Train and test conditions for decoders were determined by dichotomies identified across the task set (Supplementary Note 1). To train and test decoders, we used sklearn.svm.LinearSVC Python package. The CCGP score for a given task is the average decoding score achieved across all dichotomies where the task in question was part of either the train set or the test set. For model scores reported in the main text, we only calculate CCGP scores for models where the task in question has been held out of training. In Supplementary Fig. 9, we report scores on tasks where models have been trained on all tasks, and for models where instructions have been switched for the hold-out task.

For Fig. 3e, we calculated Pearson's r correlation coefficient between performance on held-out tasks and CCGP scores per task, as well as a P-value testing against the null hypothesis that these metrics are uncorrelated and normally distributed (using the scipy.stats.pearsonr function). Full statistical tests for CCGP scores of both RNN and embedding layers from Fig. 3f can be found in Supplementary Fig. 9. Note that transformer language models use the same set of pretrained weights among random initialization of Sensorimotor-RNNs, thus for language model layers, the Fig. 3f plots show the absolute scores of those language models.

Conditional clause/deduction task analysis

We first split our task set into two groups (listed below): tasks that included conditional clauses and simple deductive reasoning components (30 tasks) and those where instructions include simple imperatives (20 tasks). We computed the difference in performance across the mean of generalization performance for each group across random initialization for each model (Fig. 2f). We compared these differences to a null distribution constructed by performing a set of 50 random shuffles of the task set into groups of 30 and 20 tasks and computing differences in the same way, again using two-sided unequal-variance t-tests. Because STRUCUTRENET is a nonlinguistic model, we then compared performance of STRUCUTRENET to our instructed models to disassociate the effects of performing tasks with a deductive reasoning component versus processing instructions with more complicated conditional clause structure. Results of all statistical tests are reported in Supplementary Fig. 6).

Simple imperative tasks include: 'Go', 'AntiGo', 'RTGo', 'AntiRTGo', 'GoMod1', 'GoMod2', 'AntiGoMod1', 'AntiGoMod2', 'RTGoMod1', 'AntiRTGoMod2', 'RTGoMod2', 'AntiRTGoMod2', 'DM', 'AntiDM', 'MultiDM', 'AntiMultiDM', 'DMMod1', 'DMMod2', 'AntiDMMod1' and 'AntiDMMod2'.

Conditional clause/deduction tasks include: 'ConDM', 'ConAntiDM', 'Dur1', 'Dur2', 'MultiDur1', 'MultiDur2', 'AntiDur1', 'AntiDur2', 'AntiMultiDur1', 'AntiMultiDur2', 'Dur1Mod1', 'Dur1Mod2', 'Dur2Mod1', 'Dur2Mod2', 'COMP1', 'COMP2', 'MultiCOMP1', 'MultiCOMP2', 'AntiCOMP1', 'AntiCOMP2', 'AntiMultiCOMP1', 'AntiMultiCOMP2', 'COMP1Mod1', 'COMP1Mod2', 'COMP2Mod1', 'COMP2Mod2', 'DMS', 'DNMS', 'DMC' and 'DMNC'.

Language production training

Self-supervised language production network training. Our language production framework is inspired by classic sequence-to-sequence modeling using RNNs[53]. Our Production-RNN is a GRU with 256 hidden units using ReLU nonlinearities. At each step in the sequence, a set of decoder weights, $\text{Linear}_{\text{words}}$, attempts to decode the next token, $w_{\tau+1}$, from the hidden state of the recurrent units. The hidden state of the Production-RNN is initialized by concatenating the time average and maximum sensorimotor activity of a SBERTNET (L) and passing that through weights $\text{Linear}_{\text{sm}}$. The linguistic instruction used to drive the initializing sensorimotor activity is in turn used as the target set of tokens for the Production-RNN outputs. The first input to the Production-RNN is always a special start-of-sentence token, and the decoder runs until an end-of-sentence token is decoded or until input reaches a length of 30 tokens. Suppose $w_{1,k}\ldots w_{\mathcal{T},k}\in \text{Instruct}_k^i$ is the sequence of tokens in instruction k where k is in the instruction set for task i and X^i is sensory input for a trial of task i. For brevity, we denote the process by which language models embed instructions as Embed() (see 'Pretrained transformers'). The decoded token at the τth position, $\hat{w}_{\tau,k}$, is then given by

$$
\begin{aligned}
h_T^{sm} &= \text{SensorimotorRNN}\left(X^i, Embed\left(w_{1,k}\ldots w_{\mathcal{T},k}\right)\right) && h_T^{sm}\in\mathbb{R}^{T\times256}\\
sm\_out &= \left(\text{mean}_T\left(h_T^{sm}\right), \max_T\left(h_T^{sm}\right)\right) && sm\_out\in\mathbb{R}^{512}\\
\overline{h_0^{\text{decoder}}} &= \text{relu}\left(\text{Linear}_{\text{sm}}(sm\_out)\right) && \overline{h_0^{\text{decoder}}}\in\mathbb{R}^{256}\\
h_0^{\text{decoder}} &= \text{Dropout}\left(\overline{h_0^{\text{decoder}}}\right) && h_0^{\text{decoder}}\in\mathbb{R}^{256}\\
h_\tau^{\text{decoder}} &= \text{ProductionRNN}\left(\hat{w}_{1,k}\ldots\hat{w}_{\tau-1,k}; h_0^{\text{decoder}}\right), && h_\tau^{\text{decoder}}\in\mathbb{R}^{256}\\
p_{\hat{w}_{\tau,k}} &= \text{softmax}\left(\text{Linear}_{\text{words}}\left(h_{\tau,k}^{\text{decoder}}\right)\right) && p_{\hat{w}_{\tau,k}}\in\mathbb{R}^{|vocab|},\\
\hat{w}_{\tau,k} &= \text{argmax}\left(p_{\hat{w}_{\tau,k}}\right)
\end{aligned}
$$

The model parameters $\Theta^{\text{production}}=\{\text{Linear}_{\text{sm}}, \text{Linear}_{\text{words}}, \text{Production-RNN}\}$ are trained using cross-entropy loss between the $p_{\hat{w}_{\tau,i}}$ and the instruction token $w_{\tau,k}$ provided to the sensorimotor-RNN as input. We train for 80 epochs of 2,400 batches with 64 trials per batch and with task type randomly interleaved. We found that using an initial learning rate of 0.001 sometimes caused models to diverge in early phases of training, so we opted for a learning rate of 1×10^{-4}, which led to stable early training. To alleviate similar oscillation problems detected in sensorimotor training, we also decayed the learning rate by $\gamma = 0.99$ per epoch. Additionally, the use of a dropout layer with a dropout rate of 0.05 improved performance. We also used a teacher forcing curriculum, where for some ratio of training batches, we input the ground truth instruction token $w_{\tau,k}$ at each time step instead of the models decoded word $\hat{w}_{\tau,k}$. At each epoch, $\text{teacher\_forcing\_ratio} = 0.5\times\frac{80-\text{epoch}}{80}$.

Obtaining embedding layer activity using motor feedback. For a task, i, we seek to optimize a set of embedding activity vectors $E^i\in\mathbb{R}^{64}$ such that when they are input as task-identifying information, the model will perform the task in question. Crucially, we freeze all model weights $\Theta = \{\text{sensorimotor-RNN}, \text{Linear}_{\text{out}}, \text{Linear}_{\text{embedding}}\}$ and only update E^i according to the standard supervised loss on the motor output. For notional clarity, GRU dependence on the previous hidden state h_{t-1} has been made implicit in the following equations.

$$\hat{y}^i = \sigma\Big(\text{Linear}_{\text{out}}\left(\text{SensorimotorRNN}(X^i, E^i)\right)\Big)$$
$$L = \text{mMSE}(y, \hat{y})$$

We optimized a set of 25 embedding vectors for each task, again using an Adam optimizer. Here the optimization space has many suboptimal local minimums corresponding to embeddings for related tasks. Hence, we used a high initial learning rate of $lr = 0.05$, which we decayed by $\gamma = 0.8$ for each epoch. This resulted in more robust learning than lower learning rates. An epoch lasts for 800 batches with a batch length of 64, and we train for a minimum of 1 epoch or until we reach a threshold performance of 90% or 85% on 'DMC' and 'DNMC' tasks.

Producing task instructions. To produce task instructions, we simply use the set E^i as task-identifying information in the input of the sensorimotor-RNN and use the Production-RNN to output instructions based on the sensorimotor activity driven by E^i. For each task, we use the set of embedding vectors to produce 50 instructions per task. We repeat this process for each of the 5 initializations of sensorimotor-RNN, resulting in 5 distinct language production networks, and 5 distinct sets of learned embedding vectors. Reported results for each task are averaged over these 5 networks. For the confusion matrix (Fig. 5d), we report the average percentage that decoded instructions are in the training instruction set for a given task or a novel instruction. Partner model performance (Fig. 5e) for each network initialization is computed by testing each of the 4 possible partner networks and averaging over these results.

Sample sizes/randomization
No statistical methods were used to predetermine sample sizes but following ref. 18 we used five different random weight initializations per language model tested. Randomization of weights was carried out automatically in Python and PyTorch software packages. Given this automated randomization of weights, we did not use any blinding procedures in our study. No data were excluded from analyses.

Software
All simulation and data analysis was performed in Python 3.7.11. PyTorch 1.10 was used to implement and train models (this includes Adam optimizer implementation). Transformers 4.16.2 was used to implement language models and all pretrained weights for language models were taken from the Huggingface repository (https://huggingface.co/docs/transformers/). We also used scikit-learn 0.24.1 and scipy 1.7.3 to perform analyses.

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

Data availability
All weights for language transformers used in this study were taken from pretrained models available on the Huggingface repository (https://huggingface.co/docs/transformers/). Training data for simulated psychophysical tasks were generated using code available at https://github.com/ReidarRiveland/Instruct-RNN/. The full set of trained model weights for all results is available upon request.

Code availability
All code used to train models and analyze results can be found at https://github.com/ReidarRiveland/Instruct-RNN/.

References

49. Chung, J., Gulcehre, C., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. Preprint at https://arxiv.org/abs/1412.3555 (2014).
50. Radford, A. et al. Better language models and their implications. https://openai.com/blog/better-language-models/ (2019).
51. Bromley, J. et al. Signature verification using a 'siamese' time delay neural network. *Int. J. Pattern Recognit. Artif. Intell.* **7**, 669–688 (1993).
52. Wolf, T. et al. Transformers: state-of-the-art natural language processing. In *Proc. 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (eds Liu, Q. & Schlangen, D.) 38–45 (Association for Computational Linguistics, 2020).
53. Sutskever, I., Vinyals, O. & Le., Q. V. Sequence to sequence learning with neural networks. In *Proc. 27th International Conference on Neural Information Processing Systems* 3104–3112 (MIT Press, 2014).

Acknowledgements
We thank N. Rungratsameetaweemana, T. Aquino and V. Borghesani as well as N. Patel and P. Tano for their useful discussions during this project. We are also appreciative to the University of Geneva for the funding which made this research possible.

Author contributions
A.P. and R.R. conceived the project. R.R. wrote the code for model simulations and performed analysis of model representations. A.P. and R.R. wrote and revised the paper.

Funding
Open access funding provided by University of Geneva.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41593-024-01607-5.

Correspondence and requests for materials should be addressed to Reidar Riveland.

Peer review information *Nature Neuroscience* thanks Blake Richards and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

nature portfolio

Corresponding author(s):	Reidar Riveland
Last updated by author(s):	Jan 18, 2024

Reporting Summary

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

Statistics

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

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

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

Software and code

Policy information about availability of computer code

Data collection	Data was generated in simulation using Python 3.7.11. PyTorch 1.10 was used to implement and train models (this includes Adam optimizer implementation). Transformers 4.16.2 was used to implement language models and all pre-trained weights for language models were taken from Huggingface repository (https://huggingface.co/docs/transformers/). Training, analysis, and plotting scripts available at https://github.com/ReidarRiveland/Instruct-RNN.
Data analysis	Data analysis performed in Python 3.7.11 (https://github.com/ReidarRiveland/Instruct-RNN). We also used scikit-learn 0.24.1 and scipy 1.7.3 to perform analyses.

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

Data

Policy information about availability of data

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

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

All weights for language transformers used in this study were taken from pre-trained models available on the Huggingface repository (https://huggingface.co/docs/transformers/). Training data for simulated psychophysical tasks was generated using code available at https://github.com/ReidarRiveland/Instruct-RNN. The full set of trained model weights for all results included in the manuscript is available upon request.

Research involving human participants, their data, or biological material

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

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

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

Field-specific reporting

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

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

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

Life sciences study design

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

Sample size	In cases applicable we analyzed the results from models that were trained across five different random initialization of parameters. This choice was constrained by computational feasibility but is on par with other computational neuroscience studies (e.g. Yang et. al., Nature Neuroscience, 2019).
Data exclusions	None
Replication	Models tested across the five different initializations yielded consistent results throughout experiments, both in performance and in analysis of model representations. Exact random seeds are provide in code such that training and testing can be reproduced exactly.
Randomization	Again, in the case of each model we used five different random initialization. The same set of five random seeds were used across each of these conditions.
Blinding	We did not use explicit blinding in the analysis and simulation was not performed blind to conditions. This is because all experiments were executed automatically in computer code. All model weights that were not part of pre-trained language models were randomly initialized, in which case the authors had no control over these initializations.

Reporting for specific materials, systems and methods

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

Materials & experimental systems

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

Methods

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

Plants

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