A NEURAL MODEL OF TASK COMPOSITIONALITY WITH NATURAL LANGUAGE INSTRUCTIONS

A PREPRINT

Reidar Riveland
Department of Basic Neuroscience
University of Geneva
reidar.riveland@unige.ch

Alexandre Pouget
Department of Basic Neuroscience
University of Geneva
alexandre.pouget@unige.ch

February 22, 2022

ABSTRACT

We present neural models of one of humans' most astonishing cognitive feats: the ability to interpret linguistic instructions in order to perform novel tasks with just a few practice trials. Models are trained on a set of commonly studied psychophysical tasks, and receive linguistic instructions embedded by transformer architectures pre-trained on natural language processing. Our best performing models can perform an unknown task with a performance of 80% correct on average based solely on linguistic instructions (i.e. 0-shot learning), and 90% after 3 learning updates. We found that the resulting neural representations capture the semantic structure of interrelated tasks even for novel tasks, allowing for the composition of practiced skills in unseen settings. Finally, we also demonstrate how this model can generate a linguistic description of a task it has identified using motor feedback, which, when communicated to another network, leads to near perfect performance (95%). To our knowledge, this is the first experimentally testable model of how language can structure sensorimotor representations to allow for task compositionality.

Keywords Natural Language Processing · Cognitive Control · Computational Neuroscience

Introduction

In a laboratory setting, animals require numerous trials carefully presented over a period of weeks in order to acquire a new behavioral task. This is in part because the only means of communication with non-linguistic animals is simple positive and negative reinforcement signals. By contrast, it is common to give written or verbal instructions to human subjects which allows them to perform new tasks relatively quickly. For instance, a subject told to 'press the button in the direction of the stimulus presented with highest intensity', would know what to do right away. Indeed, for simple tasks like this one, human subjects would perform nearly perfectly on the very first trial (i.e. 0-shot learning), even if they had no previous experience with the particular task. Alternatively, human subjects can also learn to correctly perform a task in the absence of linguistic instructions, e.g. through positive and negative reinforcement. Remarkably, once training is completed, subjects can typically explain how they solved the task with natural language.

The dual ability to use the information in an instruction to immediately 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 and representational structures that underlie these abilities remain poorly understood. In particular, we still don't know how the human brain represents linguistic instructions in order to induce immediate sensorimotor generalization, nor do we know how sensorimotor representations are shaped by linguistic information.

One influential systems-level explanation of our ability to generalize performance posits that flexible inter-region connectivity in the pre-frontal cortex (PFC) allows for the reuse of practiced sensorimotor representations in novel settings [1, 2]. By recombining practiced stimulus-response patterns according to the ostensible demands of a previously unseen task, we can leverage well-established abilities and perform on new tasks in very few practice trials. Moreover,

various studies have shown that in instructed tasks where information about task demands is available prior to execution, activity in the PFC is prospectively biased towards the combination of sensorimotor pathways that eventually implements actions that satisfy those demands [3, 4, 5]. Recent modeling work has begun to explore the neural mechanisms of these abilities in a non-linguistic setting [6]. In a recurrent neural network (RNN) trained on a set of psychophysical tasks, the authors reported that the network could perform well on some novel tasks by approximating the new task as a linear combinations of the previously learned tasks which had related demands. This form of compositionality is encouraging but fails to capture the dynamics of a network that understands natural language instructions, where the description of a new tasks is not a simple linear combination of related rules. On the other hand, a long line of work in Reinforcement Learning (RL) has investigated how agents equipped with linguistic information can learn more rapidly and flexibly [7, 8, 9, 10]. These models display improved performance on navigation and planning tasks but often veer away from natural language instructions in favor of specialized language-like systems that are hand designed to match features in the environment. Such models are not particularly well-suited to making tractable predictions for how networks of sensorimotor neurons in the brain should be influenced by linguistic information, and indeed have never been used for that purpose.

In this work, we train RNN models on a set of commonly studied psychophysical tasks [6], where information about task demands is provided in the form of natural language instructions embedded using a pre-trained transformer architectures [11] (e.g. GPT2, BERT). These instruction embeddings are input to the RNN along with sensory stimuli, after which the model must produce the proper motor response. Combining this modeling approach with powerful pre-trained transformers allows us to make testable and interpretable predictions about the neural mechanism and representational structures that support instruction following behaviors in sensorimotor networks.

First, we show that these models can learn multiple tasks simultaneously, even using a diverse set of instruction formulations for each task. Next, we show that models can use linguistic information to immediately generalize performance to unseen tasks (0-shot learning), with an average performance of 80% correct for our best performing model. Generalization in these models is supported by the ability to organize sensorimotor activity according to abstract, task relevant axes and then leverage the semantics of instructions to immediately place activity for a novel task in the proper region of space relative to these axes. This allows the network to compose skills from practiced tasks and correctly apply them in a novel setting. This structure is also realized at the single neuron level, where tuning curves are modulated by the semantics of instructions in a principled way. Finally, we demonstrate how a network trained to interpret linguistic instructions can invert this understanding and produce linguistic instructions for a task that it has identified using only motor feedback signals. Communicating these instructions to a partner model results in a performance of 95% across tasks compared to 39% performance when directly sharing latent task representations, demonstrating the importance of natural language as an information encoding scheme that is shared between otherwise independent sets of sensorimotor neurons. To our knowledge, this is the first neural model demonstrating how a sensorimotor network can exploit the computational properties of language to generalize performance and communicate task knowledge.

Results

Instructed models and task set.

To examine how linguistic information leads to generalization across sensorimotor tasks, we train sensorimotor RNNs on a set of 16 interrelated psychophysical tasks that require various cognitive capacities which are well-studied in the literature [6]. These include perceptual decision-making [12], inhibitory control [13], matching tasks [14, 15], and combinations of these and other skills. For all tasks, models receive a sensory input and task-identifying information and must output motor response activity appropriate for the task (Fig. 1a). Input stimuli are encoded by two 1-D maps of neurons with periodic Gaussian tuning curves to angles (over $[0, 2\pi]$). As a result, a specific input value, i.e., a specific angle, is encoded by a hill of activity across the 1D maps. The two maps are meant to model two distinct sensory modalities. Output responses are encoded in the same way, via hills of activity, encoding specific output angles. Our 16 tasks were divided into 4 groups, 'Go', 'Decision Making', 'Comparison', and 'Matching', where within group tasks share similar sensory input structures but may require divergent responses. For instance, in the decision making task (DM), the network is presented with two stimuli and must respond in the direction of the stimulus with the highest contrast immediately after the extinction of the fixation point. In the anti-decision making task (Anti DM), the input is the same but the network must instead respond in the direction of the stimulus with the weakest contrast (Fig. 1b). Fig. 1b also illustrates a delayed match to stimuli (DMS) task and a delayed non-match to stimuli (DNMS) task, which also require different responses for the same input. Given the possibility of such divergent responses, 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 Info. 8 for example trials).

In our models, this task-identifying input is either non-linguistic or linguistic. For our non-linguistic control model, called *simpleNet*, the identity of a task is represented by one of 16 orthogonal rule vectors which index each task (Fig. 1a, *simpleNet*). Alternatively, *Instructed Models* use a pre-trained transformer architecture [11] to embed natural language instructions for the tasks at hand. These embeddings are then passed to the sensorimotor RNN as task-identifying information (Fig. 1a, Instructed Models). Crucially, for each task we formulate a set of 15 unique instructions, forcing instructed models to leverage the semantics shared across different formulations of descriptions for the same task, rather than keying on particular phrasings to differentiate tasks (see Supplementary Info. 7 for full instruction set). The transformers we use – GPT-2, BERT and SENTENCE-BERT (abbreviated S-BERT hereafter) – have near-identical architectures but differ in the language modeling objective used to pre-train model weights, and consequently, possess different types of linguistic knowledge. GPT-2 [16], is trained to predict the next word in a sentence given the preceding words. BERT [17] is also trained on a similar word level objective along with a simple unsupervised sentence level objective, in which the network is given two sentences and must determine whether they follow each other in the original text. S-BERT [18] is trained like BERT but receives additional supervised tuning on the Stanford Natural Language Inference task [45], a hand-labeled dataset detailing the logical relationship between two candidate sentences (see Methods). In addition to linguistic training, we also test models where we allow weights in the last three layers of the language transformer to update according to loss from the sensorimotor output ('tuned' models, see Methods for details), effecting a type of linguistic grounding [20]. Finally, as a control we also test a shallow, Bag of Words embedding scheme that only uses word count statistics to embed each instruction.

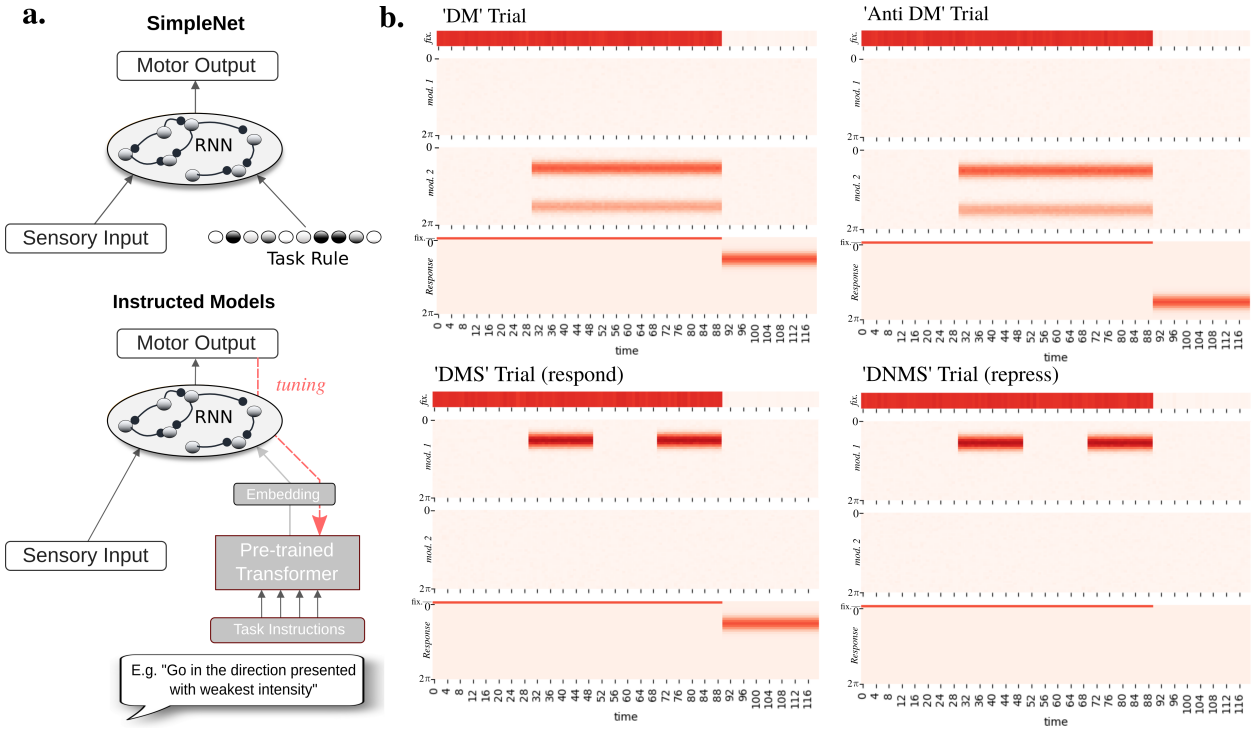

Figure 1 | Instructed Models Learn the Task Set. a. Diagram of models: *simpleNet* consists of an RNN that receives information about the current task through 1 of 16 orthogonal rule vectors indexing each task. Instructed models use an identical RNN as *simpleNet*, but receive information about task type by embedding a natural language instruction for the task using a pre-trained transformer architecture. Example instruction for the 'Anti DM' task shown. Red tuning arrow indicates gradient flow from motor loss into transformers for 'tuned' models. **b.** Example trials for 4 tasks, consisting of a fixation unit (fix.) two sensory modalities (mod.1, mod. 2) and target response. In each case, identical inputs require varying responses across different tasks, a pattern which is repeated across our task set.

All of our models can learn to perform all tasks simultaneously (Supplementary Info. Fig. 1a). For non-linguistic *simpleNet* high performance across all tasks is simply a question of the representational power of the sensorimotor RNN, as the contextual information about each task is already perfectly separated. This is more complicated for the instructed models, which are presented with 15 different instruction formulations for each task. The model must first infer the common semantic content between these 15 formulations before it can properly gate information in a task-relevant manner in the sensorimotor RNN. Transformer pre-training objectives introduce correlations between instructions that

make this additional step harder for some and easier for others. For example, GPT-2 requires many more training examples to reach asymptotic performance than S-BERT models.

To test whether our model can generalize performance to new linguistic instructions for the learned tasks, we tested all architectures on 5 new instructions per task, with no further training (Supplementary Info. Fig. 1b). Only *sbertNet-tuned* maintains our threshold performance of 95% performance on new instructions across all tasks, although all architectures exhibit robust generalization. This demonstrates that networks do not simply memorize instruction sets, but instead infer their proper semantic content even for entirely novel instructions.

Generalization to novel tasks.

We next turned to how training networks to perform tasks with natural language instructions can help those networks generalize performance to novel tasks. To this end, we used a task holdout procedure in which we trained individual networks on 14 of our 16 total cognitive tasks. We then tested network performance over an initial period of exposure to training examples from held out tasks. Results are shown in Fig. 2.

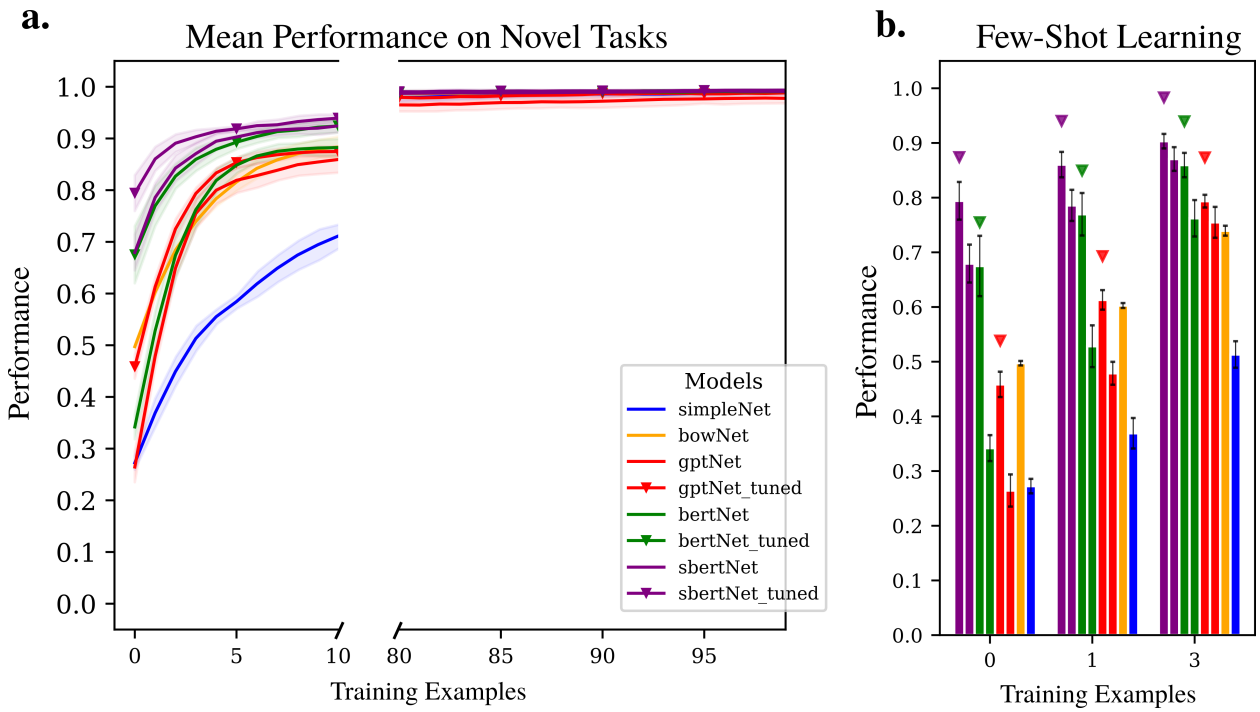

Figure 2 | Model Performance on Novel Tasks. a. Performance on holdout tasks averaged over all tasks. x-axis split to show rapid learning over the first 10 trials, then asymptotic performance after 80 training steps. **b.** Model performance averaged across holdout tasks for 0, 1, and 3 training examples.

Our uninstructed control model *simpleNet* performs at 27% on average for novel tasks. This serves as a baseline for generalizing performance, as orthogonal task rules contain no information about how tasks relate to one another. Note that chance performance is not 50%, despite the fact that many of our tasks are two-alternative forced choice. Indeed, given how we score trials and the continuous nature of motor outputs (see Methods), a random network would effectively perform at 0% correct. Therefore, 27% is not random level performance per se, but rather a reflection of the performance achieved by the default response mode of a network trained and tested on a task set with a degree of shared structure (e.g. always repressing response during fixation, always responding to a coherent angle etc).

Next we tested generalization in networks that received linguistic instructions. *gptNet*, which uses GPT-2 to embed instructions, learns the new tasks faster than *simpleNet*, but, strikingly, 0-shot performance is identical to that of *simpleNet* (Fig. 2, solid red line). In other words, *gptNet* cannot infer the demands of a novel task on the basis of instructions, despite the fact that GPT-2 possess a deep and sophisticated architecture, is extensively pre-trained on a massive corpus of text, and demonstrates a strong ability to produce coherent text given a prompt [21]. However, fine tuning the weights of the top layers of GPT-2 (resulting in *gptNet-tuned*) increases zero-shot performance (up to 46% , Fig. 2, solid red line with triangles), indicating that tuning language representations in response to the motor loss improves generalization.

bertNet, an architecture which uses a BERT transformer, only shows slightly better 0-shot performance (34%) compared to *simpleNet* and *gptNet*. However, if we fine-tune the weights of the language transformer according to the sensorimotor loss, we observe a large increase in generalization, with *bertNet-tuned* 0-shot learning now at 69% and 3-shot learning at 84%. Recall that like GPT-2, BERT is pre-trained on unsupervised word prediction but unlike GPT-2 also learns to determine whether two sentences are adjacent in a document. This suggests that the semantic representations developed by BERT through learning the relationship between successive sentences greatly aid in generalization.

Interestingly, *sbertNet*, which relies on the S-BERT, reaches the same level of 0-shot performance, 69%, with no tuning of the language transformer (Fig. 2 purple line). S-BERT is trained like BERT on word prediction and the successive sentence objective, but also benefits from additional supervised training on the Stanford Natural Language Inference task in which the network has to classify the logical relationship between sentences [45]. Tuning S-BERT transformer weights results in even stronger performance, with *sbertNet-tuned* achieving a 0-shot performance of 80% and a 3-shot performance of 90 %. Note that this is the performance averaged across all held out tasks. In many cases (e.g. 'Go', 'DM', 'Anti DM'), we observe greater than 95% 0-shot performance, which is remarkable given that the network has never been exposed to these tasks but nonetheless manages to perform them nearly perfectly based solely on linguistic instructions (see Supplementary Info. 2 for performance across tasks).

Overall, the extent to which instruction representations produced by our language transformers can aid in generalization depends on two factors. First, allowing the sensorimotor loss to update the weights in the last layers of the transformer, thereby adapting linguistic representations to the specific stimulus-response structure of the task set, improves 0-shot performance. For our Comparison group of tasks, where very different motor demands are denoted by subtle syntactic differences, tuning is essential. Models that rely on pure language pre-training often misidentify these tasks and perform incorrectly (see Supplementary Info. 3). Secondly, pre-training that endows transformers with increasingly precise knowledge of sentence-level semantics (BERT's adjacent sentence objective and the Stanford Natural Language Inference task) leads to greater generalization.

Indeed we found that the ability of our best performing instructed models to leverage the real semantic value of each instruction was crucial for generalization. To test this, we performed a task instructions swap on the held out tasks. Specifically, we presented the network with the sensory input for one held out task, while providing the linguistic instruction for the other held out task. Models that truly infer the proper sensorimotor response based on the semantic content of instructions should be most penalized by this manipulation. As predicted, our three best performing models - *bertNet-tuned*, *sbertNet*, *sbertNet-tuned* - experienced the largest decrease in performance when instructions are swapped (-45% ,-43%, -59% respectively, see Supplementary Info. 4).

Finally, to determine the impact of word ordering, which affects the syntactic and semantic structure of the instructions, we also trained a model with a simple Bag or Words embedding, *bowNet*. This model performs slightly better than *gptNet* but well below *sbertNet* and *sbertNet-tuned*, confirming that S-BERT can extract the meaning encoded by subtle syntactic cues like the word order of instructions to perform well in a 0-shot setting.

Instructed models infer abstract neural structure.

In order to investigate the neural representations that lead to immediate generalization in novel tasks, we examined hidden activity of sensorimotor units at the time of stimulus onset. Fig. 3a,b shows the first two principal components' (PCs) for each task in the 'Go' family tasks for *simpleNet* and *sbertNet-tuned*.

Plotting sensorimotor activity of *simpleNet* along the first two PC's reveals a principled relationship between task representations when the model has been exposed to all tasks during training (Fig. 3a, top). In particular, activity is organized along abstract, task relevant axes. In this case, there is a 'Pro' vs. 'Anti' direction in activity space that is consistent whether applied to 'RT' or 'Standard' versions of the 'Go' task ('RT' vs 'Standard' define another axis orthogonal to the 'Pro' vs. 'Anti' axis). Importantly, these abstract axes are no longer present when 'Anti Go' is held out of training (Fig. 3a, bottom). Indeed, the vectors relating 'Go' to 'Anti Go' and 'RT Go' to 'Anti RT Go', which are parallel in Fig. 3a, top, are now nearly orthogonal. This lack of structure predicts that *simpleNet* should achieve very low 0-shot performance on held-out 'Anti Go', which is in fact the case (1% 0-shot performance; see Supplementary Info. Fig. 1b).

Task representations for 'Go' tasks induced by *sbertNet-tuned* show a similar structure to *simpleNet* when the model is trained on all tasks (Fig. 3b, top). However, in striking contrast to *simpleNet*, *sbertNet-tuned* is also able to infer this structure among tasks even when 'Anti Go' is held out of training. The model properly places 'Anti Go' activity along the 'Anti' direction relative to 'Go' (Fig. 3b, bottom). Note that the abstract nature of the 'Anti' vs. 'Pro' axis is essential when dealing with unseen tasks. By virtue of training on 'Anti RT Go' and 'RT Go' the model already has knowledge of how sensorimotor transformations should shift as activity moves along this axis. Hence, when *sbertNet-tuned* correctly infers the relation between 'Anti Go' and 'Go', the model has the opportunity to compose subcomponents of tasks it has

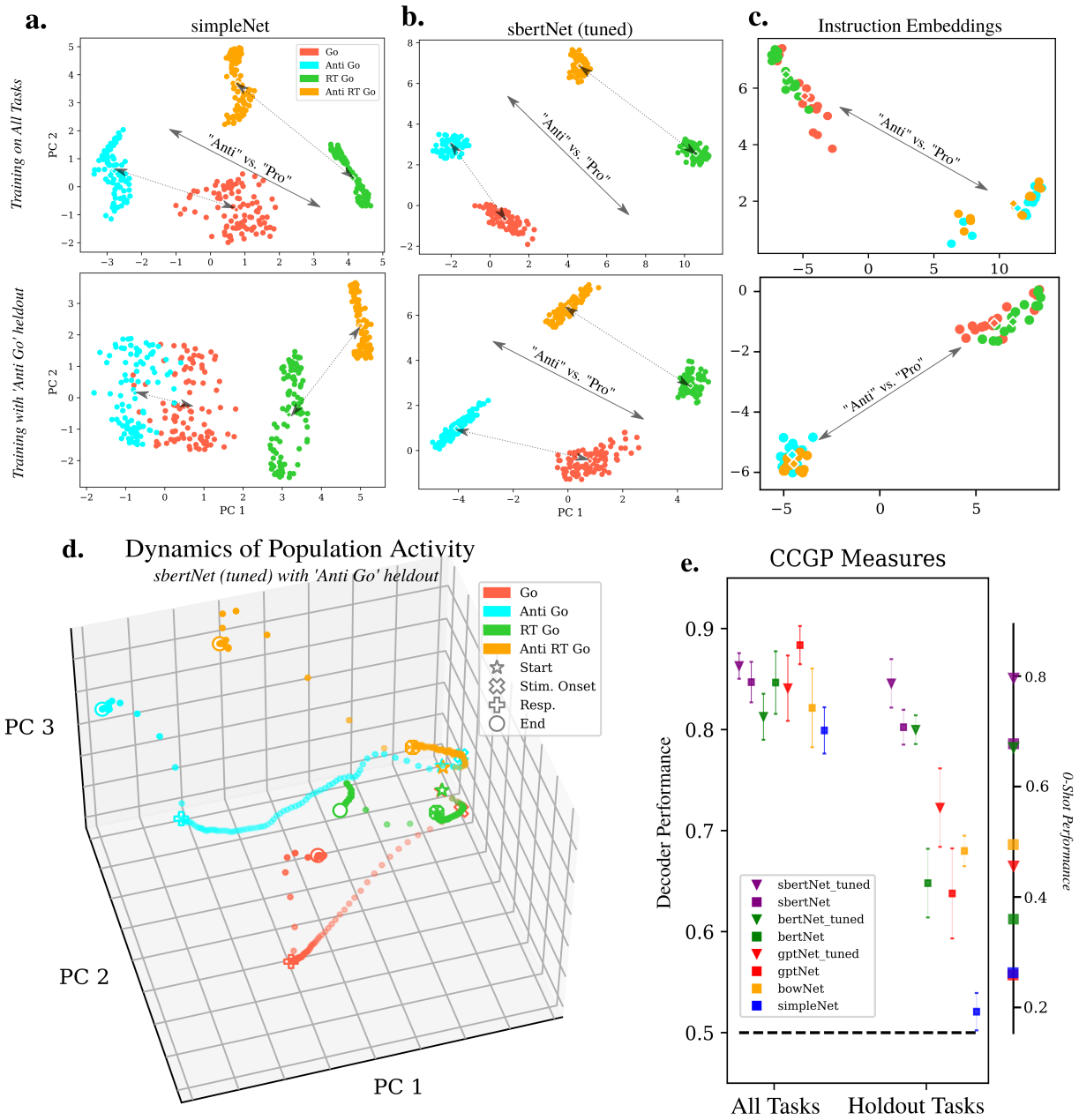

Figure 3 | Abstract Representations in Instructed Models. **a.** First two PC's of sensorimotor hidden activity in *simpleNet* for 100 example trials of each task in the 'Go' family at stimulus onset. Top: model trained on all tasks, bottom: model trained with 'Anti Go' held out. **b.** Same as **a** but for *sbertNet-tuned*. **c.** Same as **b** except for the last hidden layer of transformer instruction embeddings for each instruction in the 'Go' family of tasks. **d.** *sbertNet-tuned* hidden state trajectories for an example trial of each task in the 'Go' family, where 'Anti Go' has been held out of training. **e.** Cross Condition Generalization Performance quantifying the ability of each model to immediately infer abstract task relationships when trained on all tasks and when a task has been held out of training. Comparison with with 0-shot performance.

acquired during training and apply them to a novel setting. In the case of 'Anti Go', properly grasping the relationship between tasks along this axis leads to a 0-shot performance of 94% (see Supplementary Info. Fig. 1b). That is, without ever explicitly training on 'Anti Go', *sbertNet-tuned* can nonetheless perform the task almost perfectly.

The emergence of these structured representations in the context of an unseen task (i.e. the 0-shot setting) depends on the models ability to immediately relate unseen tasks to practiced tasks. For *simpleNet* this is an impossible undertaking,

as orthogonal rules carry no a priori information about what is common and different among tasks. Our instructed models, by contrast, have access to the richly structured information contained in the set of linguistic instructions. If we examine the last hidden layer of the transformer *sbertNet-tuned* uses to embed instructions for the 'Go' family of tasks, we see a similar 'Anti' vs. 'Pro' axis emerge (Fig. 3c, top). Again, this structure is maintained even when the model hasn't seen any 'Anti Go' instructions during the course of training (Fig. 3c, bottom). Note, again, that not all pre-training objectives embed instructions such that this structure emerges (Supplementary Info. 5). Further, disrupting the semantic relations between unseen and practiced tasks by swapping instructions destroys the relationship between task representations found in Fig. 3b, bottom, indicating that leveraging semantics is the key factor that leads to immediate structure in task representations (Supplementary Info. 4, 5).

An extensive and growing body of work holds that the computations which support behavior are embodied in the coordinated temporal dynamics of activation in the entire neural circuit [22, 23]. Specifically, recent work suggests that modulation of dynamics is the essential underlying mechanism implementing flexible sensorimotor mappings [24]. We can take a broader view of the representations discussed above and examine how semantic values of instructions modulate population dynamics throughout the duration of a trial. Fig. 3d shows the hidden state dynamics of *sbertNet-tuned* sensorimotor activity for an example trial of each task in the 'Go' family. Again, 'Anti Go' has been held out of training. Trajectories show a similar structure as observed in Fig. 3b, bottom. Importantly, these divergent trajectories emerge from an input stimulus which is presented at the *same* direction across tasks. The variation in trajectories is due to inputting different task instructions, which, in effect 'tells' the network what computation to make. Coherent semantic relations between task instructions (as shown in Fig. 3c., bottom) ensure that each computation is appropriate for the demands of the instructed task, and each trajectory plotted results in a correct response, even for 'Anti Go' which is held out of training.

To more precisely quantify the principled structure that may exist between representations of practiced and held out tasks across all tasks groups we measure the Cross Condition Generalization Performance (CCGP) of models for each group [25]. CCGP measures the ability of a linear decoder trained to differentiate one set of conditions (e.g. 'RT Go' and 'Anti RT Go') to generalize to an analogous test set of conditions (e.g. 'Go' and 'Anti Go'). Intuitively, this captures the extent to which models have learned to place sensorimotor activity along abstract task group axes (e.g. the 'Anti' dimension). Scores averaged across all task groups are shown in Fig. 3e (see Methods for details). For models that have been trained on all tasks, we observed CCGP scores at or above 80% for all model, showing that networks converge on a coherent, abstract organization of task representations when exposed to all tasks. When measuring CCGP across task groups with a task held out of training, *sbertNet-tuned* still achieves a score on par with models that have trained on all tasks. The CCGP score of all the other models drop. In the case of *simpleNet* CCGP is at 52% while performance of instructed models is tightly related to how well these models induce 0-short generalization with performance of instructed models. Furthermore, swapping instruction sets greatly reduces CCGP for our best performing models, indicating that the semantic relationship between instruction sets is essential for properly organizing the representations for unseen tasks in relation to practiced tasks (Supplementary Info Fig. 4b).

This evidence suggests that models exhibiting generalization do so by relating practiced and novel tasks along abstract axes that capture the subcomponents of each task group. Language transformers leverage the semantic relations between practiced and novel instruction sets to cue sensorimotor activity for an unseen task in the proper region relative to these axes, thereby composing practiced behaviors to perform well in a novel setting.

Semantic modulation of single unit tuning curves.

In the previous section we outlined the structured representational scheme that allows our best performing instructed models to use language to generalize performance to unseen tasks. With our artificial models we can go a step further and examine tuning profiles of individual sensorimotor neurons. As one would expect, we found that individual neurons are tuned to a variety of task relevant variables, such as the position of the input in a 'Go' task, or the difference in intensity between the input stimuli in a decision making task. Critically, however, we find neurons where this tuning is not invariant within a task group and is instead modulated by the semantic content of input instructions in a way that reflects task demands.

For instance, for the 'Go' family of tasks, unit 86 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 Anti-Go task, which was held out during training, i.e. tuning reversal is induced solely by the linguistic instructions, with no additional training.

We observed a similar behavior in the context of the Decision-Making tasks. Fig. 4b shows the activity of unit 11 with respect to Δ strength $=$ strengthA $-$ strengthB, where A and B refer to the two input stimuli. For the standard unisensory and multi-sensory decision making tasks this particular unit responds whenever Δ strength is positive, while

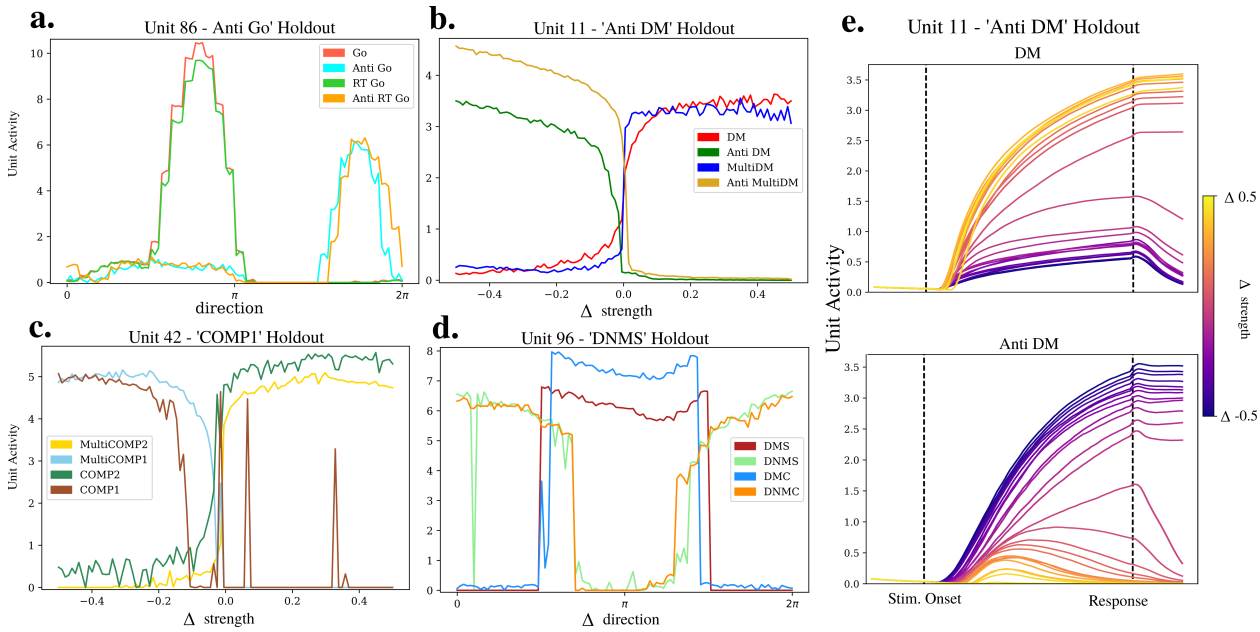

Figure 4 | Semantic Modulation of Single Unit Tuning Curves a.-d.) Tuning curves for example *sbertNet-tuned* sensorimotor units in models with various tasks held out of training recorded during the response epoch. In each case all tasks from the given task groups are shown to demonstrate how tuning profiles shift according to task demands. Note that where possible input structures are held constant, and hence modulation of unit activity is due the different semantic values of instructions for different tasks. **e)** Full activity traces for unit shown in **b** in 'DM' and 'Anti DM' tasks for different levels of relative stimulus strength.

responding to negative Δ strength for anti versions of these tasks (which requires that the network responds in the direction of the weakest stimulus). Again, this is true even for the held out task, 'Anti DM'.

Likewise, Unit 42 in the Comparison family shows a similar modulation of tuning profile by task instructions. The key variable here is Δ strength $=$ strength1 $-$ strength2, the difference in intensity between the stimulus appearing first, and the one appearing later in the trial . 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 respond to the second stimulus if the second stimulus is higher intensity, and not respond otherwise. These two tasks are particularly demanding because their instructions can only be distinguished by attending to word ordering. Despite these subtle differences in instruction sets, the language embedding can reverse the tuning of this unit in a task-appropriate manner.

Lastly, Unit 96 for the 'Matching' family of tasks encodes thresholds in difference of direction at $\pm\frac{\pi}{2}$, which is the criteria for response in category tasks. Like example neurons for other task groups, this neuron switches its tuning for matching vs. non-matching versions of the tasks which have opposing task demands. Note that for all task groups, structured tuning according to task demands is maintained even for tasks that have been held out of training. Again we emphasize that where relevant all stimulus variables remain constant, so that the variation in single unit tuning displayed in each figure is due entirely to the changes in the semantic values of input instructions.

We can also examine how the content of instructions affects the temporal evolution of neural activity for single units across the relevant trial epochs. Fig. 4e shows traces of Unit 11 activity in both the 'DM' and 'Anti DM' tasks ('Anti DM' is held out of training) for different levels of coherence. In both tasks, we observe ramping activity where the rate of ramping is relative to the strength of evidence (in this case, coherence). This motif of activity has been reported in various previous studies [26, 27]. However, in our models we find, for the first time, that an evidence accumulating neuron can swap the sign of its integration in response to a change in linguistic instructions. This change of sign between 'DM' and 'Anti DM' tasks reflects the diametric relationship between conditions for response between the two tasks, and is therefore integral for performing the task correctly. Moreover, this modulation of neural tuning is accomplished in a 0-shot setting, when the pre-synaptic weights of the neuron have never explicitly tuned to the 'Anti DM' task. This result predicts the existence of neurons in the human brain which are similarly modulated by the semantics of instructions in order to support sensorimotor generalization at the single unit level.

Linguistic communication between networks.

So far we have offered an account of what representations emerge in a network that understands the sensorimotor implications of linguistic instructions. 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 invert the language-to-sensorimotor mapping our models learn during training so that they can provide a linguistic description of a task based on the state of sensorimotor units. We then pose the following problem to our networks: after training, suppose the network is tested on a sequence of trials in which it is presented with a sensory input and the corresponding motor output for a known task, but without the linguistic instructions that normally provide essential task identifying information. Our goal is now to determine whether our instructed models can use the motor feedback it observes to subsequently provide an appropriate linguistic instruction for the current task and whether these instructions can effectively guide the performance of another network.

We begin with *sbertNet-tuned* models trained to perform all tasks. Next we construct an output channel, which we call the language production network (Fig. 5a,b; blue), to map the information contained in sensorimotor RNN to the linguistic instructions. To train this language production network, we used a self-supervised procedure: during initial training, when the network is presented with sensory inputs and a linguistic description of the task, these instructions are used as targets to train the language production network (Fig. 5a, see Methods for training details).

Once we have trained this language production network, we test its utility in communicating information acquired exclusively in a sensorimotor modality. To that end, we present the network with a series of example trials for a specific task along with motor feedback, but withhold the linguistic instructions. We then randomly initialize activity in the instruction embedding layer ('Embedding' in Fig. 5b) and update this activity in order to reduce the error between the output of the model and the motor feedback signal, while keeping all other model weights frozen. Once the activity in the instructions embedding layer drives sensorimotor units to achieve a performance criteria (at least 95% correct performance) over the series of example trials, we use the language production network to decode a linguistic description of the current task from the sensorimotor activity in the RNN. All following results use instructions decoded in this manner. This training procedure could also be implemented by using feedback connections from the RNN to the instruction embedding layer, as is done in predictive coding network [28].

To quantify the accuracy of the generated instructions, we first plot a confusion matrix (Fig. 5c). Each entry in the confusion matrix indicates whether the decoded instruction was included in one of the instruction sets used during self-supervised training. Crucially, the 'novel' column indicates a decoded instruction that was not included in the training set for the language production network (see Methods for details). Overall, these novel instructions make up 55% of all decoded instructions (see Supp. Info 9 for decoded novel instructions). Of course, the novelty of these instructions says nothing about how well they convey information about the current task and in the end are only useful to the extent that they are usable by another agent to achieve good performance. To test this, we take another *sbertNet-tuned* agent pre-trained to perform all tasks and examine this partner model's performance across tasks with instructions generated by the first network. Results are shown in Fig. 5d (blue bars). Partner model performance is 95% on average across tasks, demonstrating that generated instructions contain the appropriate semantics for guiding partner responses in each task. Importantly, this information is present even for 'novel' instructions, which are also previously unseen by the partner model (average performance across these instructions is 91%). Finally, if we test partner models where tasks have been held out of training, we find that decoded instructions guide partner models to a 0-shot performance of 75% across held out tasks compared to 80% for experimenter provided instructions. This indicates that, rather than simply memorizing the training set, self-supervised learning has endowed the language production network with a simple but effective generative model mapping sensorimotor representations to linguistic descriptions with proper semantics.

Nonetheless, our model has one important limitation: it cannot currently generate valid instructions for held out tasks (held-out for the instructions producing model as opposed to the partner network). This is probably because our language production network is small (56 thousand parameters) and has been trained only on a limited set of instructions (240 expressions). It is quite likely that much larger networks and training sets are required to generate the semantic representations that can generalize to held out tasks. We note indeed that S-BERT contains 177 million parameters that have been tuned to a massive data set (3.3 billion words).

It remains the case that, with the language production network, our models can effectively communicate information gained in a sensorimotor modality with natural language. However, given that the instructing and partner models share the same architecture, one would expect that it is more efficient for the two networks to use direct neural communication, instead of language-based communication. For instance, we can forego the language component of communication altogether and simply copy the embedding inferred by one model into the input of the partner model. Strikingly, this results in only 39% correct performance on average, compared to 95% for linguistic communication (Fig. 5d, grey bars), and only 29% performance when testing partner networks on tasks that have been held out of training. The reason for this large drop off is straightforward: the two networks share the same architecture but not the same weights.

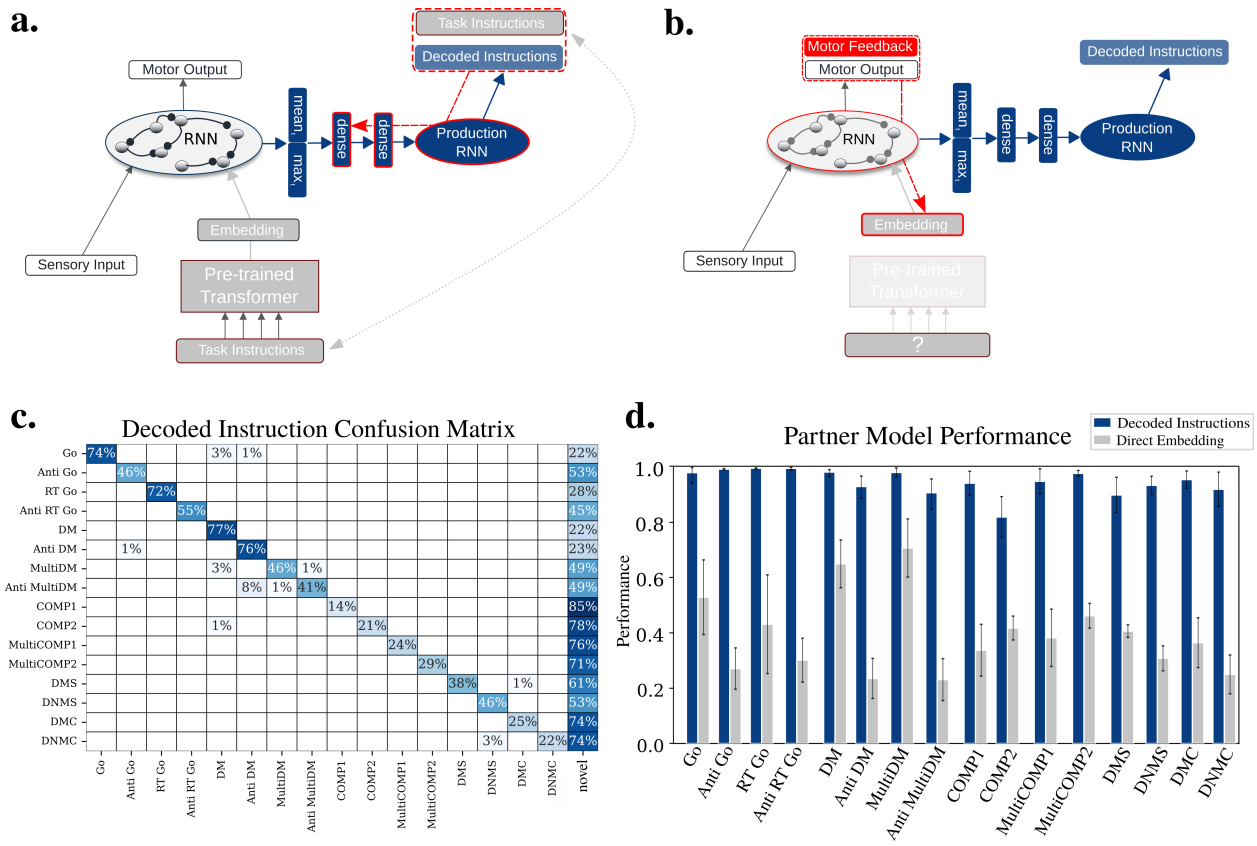

Figure 5 | Communication Between Networks a. Illustration of self-supervised training procedure for the linguistic production network (blue). Red dotted line indicates gradient flow. **b.** Illustration of motor feedback training used to spontaneously produce task descriptions in the absence of linguistic instructions. **c.** Confusion matrix of instructions decoded from sensorimotor activity evoked by embeddings inferred in **b**. **d.** Performance of a partner model trained on all tasks given decoded instructions or given direct input of embeddings.

Because of differences in initial conditions, course of training, etc. the same competencies are realized by a different set of weights in each model. Accordingly, using a neural representation tuned for one set of weights won't necessarily produce good performance in the other.

This result highlights the major advantages of linguistic communication. Our network can compress the information it has gained through experience and transfer that knowledge to a partner network such that the partner can forego this learning process and simply follow a given instruction to perform the task. Crucially, however, in order to effectively communicate that information it needs to pass through a representational medium that is equally interpretable for both sets of independent network weights. In humans and for our best performing instructed models, this medium is language.

Discussion

We have shown that recurrent neural networks can learn to perform a set of psychophysical tasks simultaneously by using a pre-trained language transformer to embed a natural language instruction for the current. Remarkably, our best performing models can leverage these embeddings to perform a brand new task on the very first trial, with an average performance of up to 80% correct based solely on a linguistic description of the task. This generalization is supported by properly inferring the structured relations between tasks at both the population and single neuron level, and this inference is in turn supported by language models' ability to properly relate the semantics of task instructions. Finally, once a network understands how to interpret instructions to perform a sensorimotor task, it can invert this process and provide a linguistic description of the task based on the sensory-motor contingency it observes, in the absence of any linguistic input. We find that communicating through language, as opposed to via directly sharing neural representations, is essential for effectively passing information between independent sets of sensorimotor neurons.

To our knowledge, this is the first account of how linguistic understanding and communication can extend both the generality and transferability of knowledge in a network of sensorimotor neurons.

A series of work in reinforcement learning has investigated using language and language-like schemes to aid agent performance. Agents receive language information in a variety of ways such as through step-by-step descriptions of action sequences [8, 9, 29, 7, 30], learning policies conditioned on a language goal [31, 32, 33, 34], by shaping a reward function given language inputs [35, 36] or by leveraging the predictive power of pre-trained models like GPT to aid directly in planning and action selection [10, 37]. It is unclear whether these models leverage the full power of natural language instructions. Agents often receive linguistic inputs that are parsed or simply refer to environmental objects, or conversely output actions in the form of a language which is hand coded to interface with the action space of the agent's virtual environment. In contrast, our model learns the mapping from language to action from scratch. On the other hand, some larger versions of the pre-trained language models we use to embed instructions display instruction following behavior in their own domains e.g. GPT-3 [38] and InstructGPT [39] in the modality of language and DALL-E [40] in a language to image modality. The semantic and syntactic understanding displayed in these models are striking. However, these models do not shed any light on the dynamics of a sensorimotor system tuned to interpret language. DALL-E's cross modal abilities does some work in this direction, showing visual understanding of potentially counterfactual scenes described by language, but it has yet to be shown that the model uses language as a genuine instruction as opposed to merely a prompt. We don't know, for example, if it has the ability displayed in our models to parse syntactically complicated instructions, for example: 'draw a dog and if the dog is brown draw a cat otherwise don't draw a cat.' Unfortunately, we couldn't test these questions because the weights of GPT-3 and DALL-E are not open source. In all, none of these models offer a representational account of *how* language might be used to induce generalization and task compositionality in the brain.

As large pre-trained transformers become more popular as neural models of human language processing [41], the failure cases we find are as important as our successful models. In particular, the inability of GPT-2 to induce 0-shot learning beyond the non-linguistic *simpleNet* architecture, suggests that training on next word prediction is not sufficient to engender the kind of semantic representations required for immediate generalization. Perhaps, GPT-3, a 175 billion parameter version of GPT-2, might fare better, however, we note that despite the massive number of parameters, GPT-3 still struggles with the type of natural language inference tasks that are an essential part of pre-training for our best-performing *sbertNet* and *sbertNet-tuned* models [38]. Yet, GPT-like architectures are most successful at predicting fMRI and ECOG data from classic language networks when participants read sentences [41], suggesting that they share fundamental similarities with language processing in the human brain. This leaves the question of how best to model the full range of human language capacity with large pre-trained transformers in a middle ground: GPT-like architectures clearly capture something important about human language processing, potentially because both the brain and GPT are essentially predictive, yet our results show that word-level prediction by itself fails to build language representations that are key to structuring sensorimotor representations which generalize to unseen settings. Instead, our findings show that some notion of sentence level semantics is key for robust 0-shot generalization. Therefore, a confluence of the sentence-level semantics within an underlying word-level prediction objective may offer a fruitful path forward in predicting neural data in language brain areas and understanding how language based semantics can modulate neural processing in the rest of the brain.

Our findings also mesh with a long series of results which demonstrate that some structured geometry organizes neural representations across many domains [25, 42, 43], and that the compositionality enabled by this organization is essential for cognitive flexibility [1, 6, 44, 2]. Agents using a compositional coding scheme must accomplish two distinct but related learning goals. Firstly, agents must learn the proper combination of abstract subcomponents for a given task. Models of compositional systems often focus on this question, choosing to start with a set of compositional building blocks (algebraic task rules or straightforward combinations of simple word cues) in order to study how agents learn to combine these components. But even before this recombination can occur, agents must construct this set of building blocks i.e. decompose the task into abstract subcomponents, often based on a learning signal that is agnostic with respect to what might be a useful abstraction to make. Our results from CCGP measures indicate that both instructed models and our non-linguistic *simpleNet* can indeed organize task representations along abstract task axes (Fig. 3e, all tasks). The key difference is that in *simpleNet* this compositional structure emerges only because of the relationship between sensorimotor mappings denoted by each task rule, resulting in little generalization to novel tasks, whereas in our best performing instructed models the subcomponents of these sensorimotor mappings are scaffolded according to the latent axes of semantic space produced by transformer instruction embeddings. Decomposing sensorimotor transformations according to a latent semantic space is the key representational scheme that makes generalization work, as shifting activity in the semantic space corresponds to an analogous shift in a sensorimotor space, leading to instruction following behavior.

The predictions our models make for population and single unit representations required to support generalization can guide future experimental work examining the interplay of linguistic and sensorimotor skills in humans. This theoretical

input may be especially useful to interpret high resolution imaging data and multi-unit recordings in humans, our only example of organisms with natural linguistic competencies. Many cognitive competencies are as of yet unaddressed by our task set and our instruction sets fall short of capturing every aspect of complexity exhibited in the full range of human language. However, by developing models that can both understand instructions as guiding a particular sensorimotor response, and communicate the results of sensorimotor learning as an interpretable linguistic instruction, we believe we have given a first demonstration of the power of language in encoding and transferring knowledge in networks of sensorimotor neurons.

References

[1] Michael W Cole, Jeremy R Reynolds, Jonathan D Power, Grega Repovs, Alan Anticevic, and Todd S Braver. Multi-task connectivity reveals flexible hubs for adaptive task control. *Nature neuroscience*, 16(9):1348–1355, 2013.

[2] Takuya Ito, Guangyu Robert Yang, Patryk Laurent, Douglas H Schultz, and Michael W Cole. Constructing neural network models from brain data reveals representational transformations linked to adaptive behavior. *Nature Communications*, 13(1):1–16, 2022.

[3] Earl K. Miller and Jonathan D. Cohen. An integrative theory of prefrontal cortex function. *Annual Review of Neuroscience*, 24(1):167–202, 2001. PMID: 11283309.

[4] Michael W Cole, Todd S Braver, and Nachshon Meiran. The task novelty paradox: Flexible control of inflexible neural pathways during rapid instructed task learning. *Neuroscience & Biobehavioral Reviews*, 81:4–15, 2017.

[5] Hannes Ruge, Theo AJ Schäfer, Katharina Zwosta, Holger Mohr, and Uta Wolfensteller. Neural representation of newly instructed rule identities during early implementation trials. *Elife*, 8:e48293, 2019.

[6] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

[7] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. *CoRR*, abs/1706.07230, 2017.

[8] David Chen and Raymond Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pages 859–865, 2011.

[9] Junhyuk Oh, Satinder P. Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. *CoRR*, abs/1706.05064, 2017.

[10] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[12] Joshua I Gold and Michael N Shadlen. The neural basis of decision making. *Annu. Rev. Neurosci.*, 30:535–574, 2007.

[13] Douglas P Munoz and Stefan Everling. Look away: the anti-saccade task and the voluntary control of eye movement. *Nature Reviews Neuroscience*, 5(3):218–228, 2004.

[14] Earl K Miller, Cynthia A Erickson, and Robert Desimone. Neural mechanisms of visual working memory in prefrontal cortex of the macaque. *Journal of neuroscience*, 16(16):5154–5167, 1996.

[15] David J Freedman and John A Assad. Neuronal mechanisms of visual categorization: an abstract view on decision making. *Annual review of neuroscience*, 39:129–147, 2016.

[16] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[18] Nils Reimers Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[19] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *CoRR*, abs/1508.05326, 2015.

[20] Martin H Fischer and Rolf A Zwaan. Embodied language: A review of the role of the motor system in language comprehension. *Quarterly journal of experimental psychology*, 61(6):825–850, 2008.

[21] Dario Amodei Daniela Amodei Jack Clark Miles Brundage Ilya Sutskever Alec Radford, Jeffrey Wu. Better language models and their implications, 2019.

[22] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43:249–275, 2020.

[23] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

[24] Evan D Remington, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, 98(5):1005–1019, 2018.

[25] Silvia Bernardi, Marcus K Benna, Mattia Rigotti, Jérôme Munuera, Stefano Fusi, and C Daniel Salzman. The geometry of abstraction in the hippocampus and prefrontal cortex. *Cell*, 183(4):954–967, 2020.

[26] Michael N Shadlen and William T Newsome. Neural basis of a perceptual decision in the parietal cortex (area lip) of the rhesus monkey. *Journal of neurophysiology*, 86(4):1916–1936, 2001.

[27] Alexander C Huk and Michael N Shadlen. Neural activity in macaque parietal cortex reflects temporal integration of visual motion signals during perceptual decision making. *Journal of Neuroscience*, 25(45):10420–10436, 2005.

[28] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[29] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. *CoRR*, abs/1912.01734, 2019.

[30] Howard Chen, Alane Suhr, Dipendra Kumar Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. *CoRR*, abs/1811.12354, 2018.

[31] Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. RTFM: generalising to novel environment dynamics via reading. *CoRR*, abs/1910.08210, 2019.

[32] Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. *CoRR*, abs/2110.01517, 2021.

[33] Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *CoRR*, abs/1906.07343, 2019.

[34] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. *CoRR*, abs/1611.01796, 2016.

[35] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018.

[36] Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. Using natural language for reward shaping in reinforcement learning. *CoRR*, abs/1903.02020, 2019.

[37] Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making, 2022.

[38] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[39] Xu Jiang Diogo Almeida Carroll L. Wainwright Pamela Mishkin Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens Amanda Askell Peter Welinder Paul Christiano Jan Leike Ryan Lowe Long Ouyang, Jeff Wu. Training language models to follow instructions with human feedback. https://cdn.openai.com/papers/Training_language_models_to_follow_instructions_with_human_feedback.pdf.

[40] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *CoRR*, abs/2102.12092, 2021.

[41] Martin Schrimpf, Idan Asher Blank, Greta Tuckute, Carina Kauf, Eghbal A. Hosseini, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. The neural architecture of language: Integrative modeling converges on predictive processing. *Proceedings of the National Academy of Sciences*, 118(45), 2021.

[42] Matthew F Panichello and Timothy J Buschman. Shared mechanisms underlie the control of working memory and attention. *Nature*, 592(7855):601–605, 2021.

[43] Edward H Nieh, Manuel Schottdorf, Nicolas W Freeman, Ryan J Low, Sam Lewallen, Sue Ann Koay, Lucas Pinto, Jeffrey L Gauthier, Carlos D Brody, and David W Tank. Geometry of abstract learned knowledge in the hippocampus. *Nature*, 595(7865):80–84, 2021.

[44] Carlo Reverberi, Kai Görgen, and John-Dylan Haynes. Compositionality of rule representations in human prefrontal cortex. *Cerebral cortex*, 22(6):1237–1246, 2012.

Methods

Model Architecture

Sensorimotor RNN

The basic model architecture and task structure used in this paper follows [6] closely. All networks of sensorimotor units, denoted SensorimotorRNN are Gated Recurrent Units [47] using ReLU non-linearities with 128 hidden units each. Broadly, inputs to the networks are made up of two components 1) sensory inputs, X and 2) task identifying information, I. We initialize hidden activity in the GRU as $h^0 \in \mathbb{R}^{128}$ with values set to 0.1. All networks of sensorimotor units use the same initialization, so we omit h^0 in network equations. At each time step, a read out layer decodes motor activity, y_t, from the recurrent activity hidden units, h_t., according to:

$$h_t = \text{SensorimotorRNN}(X_t, I_t; h_{t-1}) \qquad h_t \in \mathbb{R}^{128}$$
$$\hat{y}_t = \sigma(\text{Linear}_{\text{out}}(h_t)) \qquad y_t \in \mathbb{R}^{33}$$

where σ denotes the sigmoid function. Sensory inputs X_t are made up of 3 channels, two sensory modalities $x_{mod1,t}$ and $x_{mod2,t}$, and a fixation channel $x_{fix,t}$. Each of these inputs are continuously valued. Both $x_{mod1,t}, x_{mod2,t} \in \mathbb{R}^{32}$ and stimuli in these modalities are represented as hills of activity with peaks determined by units' preferred directions around a one-dimensional circular variable. Preferred directions for each of the 32 inputs units are evenly spaced between 0 and 2π. For an input at direction θ, the activity of a given input unit u_i with preferred direction θ_i is

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

where, str is the coefficient describing stimulus strength. The fixation channel $x_{fix,t} \in \mathbb{R}^1$ is a single unit simulating a fixation cue for the network. In all, sensory input $X_t = (x_{mod1,t}, x_{mod2,t}, x_{fix,t}) \in \mathbb{R}^{65}$. At each time step noise is added independently to each sensory input unit according to $X_{noise,t} = \sigma_{sen.}\mathbf{N}(\mathbf{0}, \mathbf{1})$. In simulations we use a value of $\sigma_{sen.} = 0.05$. Accordingly, motor output, 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 $y_t \in \mathbb{R}^{33}$

For all models, task identifying information, i.e., the task rule layer for *simpleNet* and the instructions embedding layer for the instructed model (Fig. 1a), $I_t \in \mathbb{R}^{20}$. Further, throughout the duration of a trial task identifying information remains constant and persistently active such that $I_t = I \ \ \forall t$. For all models, task identifying info I_t and sensory input X_t are concatenated as inputs to the SensorimotorRNN, meaning that the input dimensionality of all recurrent networks is 85. For our non-linguistic model *simpleNet*, we generate set of orthogonal task rules by constructing an orthogonal matrix using the Python package scipy.stats.ortho_group, and assigning rows of this matrix to each task type. This results in a set of 16 orthogonal 20-dimensional vectors uniquely associated with each task type, which are used by *simpleNet* as task identifying information in our simulations. For our instructed models, task identifying information is computed by embedding natural language instructions.

Language Modules

The main language modules we test use pre-trained transformer architectures to produce I i.e. a fixed length embedding of natural language instructions. Transformers process language inputs token by token, so in order to get a fixed length embedding we use average pooling over the sequence length in the transformer's last hidden layer. Suppose we have an input instruction $w_1 \dots w_{\mathcal{T}}$, then

$$
\begin{aligned}
h^{tran.} &= \text{transformer}(w_1 \dots w_{\mathcal{T}}), & h^{tran.} &\in \mathbb{R}^{768\times\mathcal{T}} \\
h^{I} &= \text{mean}(h^{tran.}), & h^{I} &\in \mathbb{R}^{768} \\
I &= \text{relu}(\text{Linear}_{\text{embed}}(h^{I})), & I &\in \mathbb{R}^{20}
\end{aligned}
$$

Note that the 768 dimensionality comes from the inherent model dimension of pre-trained transformers, and is shared across all our transformer based models. We also use a Bag of Words embedding to test a comparatively simple language processing scheme. Here, 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 vocabulary word in the current instruction. For our instruction set $|vocab| = 119$. This vector is then projected through a non-linearity into 20-dimensional space.

$$
\begin{aligned}
h_i^{\text{BoW}} &= \begin{cases} 1 & \text{if } \; w_i \in (w_1 \dots w_{w\mathcal{T}}) \\ 0 & \text{otherwise} \end{cases}, & h^{\text{BoW}} &\in \mathbb{R}^{|vocab|} \\
I &= \text{relu}(\text{Linear}_{\text{embed}}(h^{\text{BoW}})), & I &\in \mathbb{R}^{20}
\end{aligned}
$$

Pre-trained Transformers

We test three types of pre-trained transformer language models, GPT-2 [16], BERT [17], and S-BERT (S-BERT) [18], to embed instructions. All transformers have the same core architecture, consisting of 12 self-attention layers with an inherent model dimensionality of 768. Transformers differ in the type of pre-training objective used to tune the model parameters and therefore also differ in the types of linguistic representations each model possesses. GPT-2 is trained on a traditional language modeling objective i.e. predict next word given a context of words [16]. This model is fully autoregressive. BERT, by contrast, takes bi-directional language inputs (future and past words) 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. This endows BERT with some rudimentary notion of sentence level semantics. Extending this principle, S-BERT is explicitly trained to produce fixed length embeddings of whole sentences [18]. It takes pre-trained BERT networks and uses them in a siamese architecture [46] which allows the weights of the model to be tuned in a supervised fashion according to the Stanford Natural Language Inference dataset [45]. 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. The sentence representations produced by these networks have been shown to match human judgements of sentence similarity across several datasets. All models are 117M parameter PyTorch implementations taken from Huggingface library [49].

For each different pre-trained transformer, we also test a model where we allow the motor loss to fine tune the last layers of the transformer (details below). In all this gives us 8 types of models, *simpleNet*, which uses orthogonal rule vectors, *gptNet* and *gptNet-tuned*, (which use GPT-2 and a tuned version GPT-2 to embed instructions), *bertNet* and *bertNet-tuned*, and *sbertNet* and *sbertNet-tuned* (which use the corresponding language transformers). For each model type, we obtain results by training and testing 5 times using 5 distinct random initializations. Error bars in results show +/- one standard deviation across model seeds.

Tasks Sets

Task structure and many of the tasks themselves are identical to the ones used in [6]. Tasks were divided into 4 interrelated subgroups of 4 tasks each: the 'Go' group, the 'Decision-Making' group, the 'Matching' group, and the 'Comparison' group. Each group possesses its own internal structure which relates tasks to one another. However, all tasks can be divided into a preparatory, stimulus, and response epoch. A trial lasts for a total of $T = 120$ times steps and duration of each epoch is drawn from a uniform distribution, such that the ratio of preparatory, stimulus, and response epochs is roughly 1/5 to 3/5 to 1/5 of time in the trial. Depending on the task, multiple stimuli may appear over the course of the stimulus epoch. Also depending on the task, models may be required to respond in a particular direction or repress response altogether. Often stimuli directions are drawn from a continuous range of angles. For example trials of each task see Supplementary Info. 8.

'Go' Tasks

The 'Go' family of tasks includes 'Go', 'RT Go', 'Anti Go', 'Anti RT Go.' In both the 'Go' and 'Anti Go' 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, i.e. $\theta_{\text{stim}} \sim \mathbb{U}[0, 2\pi]$. The strength of the stimulus is given by $\text{str}_{\text{stim}} \sim \mathbb{U}[1.0, 1.2]$. In the 'Go' task the target response is in the same direction as the presented stimulus $\theta_{\text{stim}} = \theta_{\text{target}}$, while in the 'Anti Go' task the direction of the response should be in the opposite of the stimulus direction, $\theta_{\text{stim}} + \pi = \theta_{\text{target}}$. 'RT Go' and 'Anti RT Go' are reaction time versions of the standard tasks described above. Here stimuli appear for the first time in the response epoch meaning models must respond as quickly as possible. For all of these tasks, stimuli appear in only one of the two input modalities chosen with equal probability.

'Decision-Making' Tasks

The 'Decision-Making' family of tasks includes 'DM' (decision-making), 'Anti DM', 'MultiDM' (multi-sensory decision making') and 'Anti MultiDM.' For both 'DM' and 'Anti DM', two stimuli are presented simultaneously and persist throughout the duration of the stimulus epoch. They are drawn according to $\theta_{\text{stim1}} \sim \mathbb{U}[0, 2\pi]$ and $\theta_{\text{stim2}} \sim \mathbb{U}[\theta_{\text{stim1}} - 0.4\pi, \theta_{\text{stim1}} + 0.4\pi]$. A base strength applied to both stimuli is drawn such that $\text{str}_{base} \sim \mathbb{U}[0.8, 1.2]$. A coherence is drawn from a discrete distribution such that $c \sim \{-0.2, -0.15, -0.1, 0.1, 0.15, 0.2\}$ so the stimulus strength associated with each direction in a trial are given by $\text{str}_{\text{stim1}} = \text{str}_{base} + c$ and $\text{str}_{\text{stim2}} = \text{str}_{base} - c$.

For the 'DM' task,

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

and for the the 'Anti DM' task,

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

For the multi-sensory versions of each task, stimuli directions are generated in the same manner but presented over both input modalities. Coherences for both modalities are drawn independently, and criteria for target response are measured as the average strength of a given direction over both modalities.

'Comparison' Tasks

Our comparison task group includes 'COMP1', 'COMP2', 'MultiCOMP1', 'MultiCOMP2'. This group of tasks is designed to extend the basic decision-making framework into a setting with more complex control demands (this also allows us to test our models on more complicated instruction syntax, see Supplementary Info. 7). For 'COMP1' and 'COMP2', input stimuli are produce in an identical way as the in the 'DM' group of tasks, with an initial $str_{base} \sim \mathbb{U}[1.3, 1.5]$. Here, however, stim1 is presented at the beginning of the stimulus epoch, and stim2 is presented momentarily alongside stim1 after a delay. Afterwards stim1 disappears and only stim2 is displayed until the end of the stimulus epoch (see Supplementary Info. 8 for example trials). This provides a temporal ordering on the stimuli. Hence, in 'COMP1' the model must respond to the first stimuli only if has greater strength than the second and otherwise repress a response i.e.

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

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

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

'MultiCOMP1' and 'MultiCOM2' are multi-sensory versions of the above tasks and work in an analogous way as in the 'DM' family. Timing and direction of stimuli in both modalities are identical with coherence in both modalities drawn independently. Criteria for target direction are computed as the average strength of a stimulus direction across modalities.

'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), '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 strength for all stimuli use a standard strength of 1. 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 $\theta_{\text{stim2}} \sim \mathbb{U}[\theta_{\text{stim1}} - 0.4\pi, \theta_{\text{stim1}} + 0.4\pi]$. For 'DMS', models must respond to 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 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 $y_1 = y_{fix}$ during the stimulus epoch, and then either responding in the correct direction or repressing activity in the remaining target response units $y_{2 \ldots 33}$ in the response epoch. Since the model should maintain fixation until response, target for fixation is set at $y_{fix} = 0.85$ during preparatory and stimulus epochs and $y_{fix} = 0.05$ in the response epoch. When a response is not required, as in the preparatory and response 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,

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

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 i.e. $\hat{y}_{fix} > 0.5$ 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

During model training, 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.

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

Here $*$ denotes element-wise multiplication. Masks weight the importance of different trial epochs. During stimulus presentation mask weights are set to 1, during the first five time steps of the response epoch mask is set to 0, and during the remainder of the response epoch mask 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 $\Theta = \{\text{SensorimotorRNN}, \text{Linear}_{\text{out}}\}$. For instructed models we additionally update $\text{Linear}_{\text{embed}}$ in the process of normal training. We train models using standard PyTorch machinery and an Adam optimizer. For all models we use an initial learning rate $lr = 0.001$ and decay the learning rate for each epoch by a factor of $\gamma = 0.95$. An epoch consists of 500 mini-batches, with each mini-batch consisting of 128 trials. Task type remains consistent within batches, but in the case of instructed models, the instruction used to inform each trial is drawn at random across the 128 trials in the batch. Tasks are randomly interleaved during training so that the networks can learn all tasks simultaneously. During each new epoch the order of batches are shuffled to vary the order of task presentation across training. For learning all tasks we train models for 55 epochs. For holdout training, models train on 14 of the 16 tasks for 35 epochs. In each case models learn to perform all of the tasks in the training set up to a threshold of 95% correct.

Transformer Fine-Tuning

The pre-trained transformer architectures we use to embed task instructions are trained exclusively using a large corpus of natural language text according to wholly linguistic loss functions. In order to investigate the possibility that instruction embeddings may need to be grounded in the particular requirements of the prevailing sensorimotor task, we fine-tune transformer language models by allowing the gradient from the motor loss to update the weights in the final transformer layers. We do this for a relatively short period at the end of training. Specifically, during normal training we checkpoint a copy of our instructed models 10 epochs prior to the end of training. We then add the last 3 transformer layers to the set of trainable parameters, and reset the learning rates to $lr = 1e^{-4}$ for $\Theta = \{\text{SensorimotorRNN}, \text{Linear}_{\text{out}}\}$ and $lr^{\text{lang}} = 1e^{-5}$ for $\Theta^{\text{lang}} = \{\text{Linear}_{\text{embed}}, \text{transformer}_{10,11,12}\}$. Models train until they reach a threshold performance of 95% across training tasks. All tuned models achieve this mark within 10 training epochs.

Holdout Testing

During holdout testing, we present models with 100 batches of one of the tasks that has been held out of training. For instructed model the only weights allowed to update during this phase are $\Theta = \{\text{SensorimotorRNN}, \text{Linear}_{\text{out}}, \text{Linear}_{\text{embed}}\}$. All weights of *simpleNet* are trainable in this context. We use a batch size of 256 and a learning rate of $7e - 4$.

Cross Condition Generalization Performance Calculation

To calculate Cross Conditional Generalization Performance (CCGP) we examine the relations that exist between tasks within the same group. Within a task group, we record activity at the time of stimulus onset for 100 example trials of each task. We then train linear decoders to classify task activity according to one set of conditions, and subsequently test the same classifier on the remaining set of conditions. Train and test conditions are determined by dichotomies identified within each task group (see Supplemental Info. 6 for all dichotomies). As an example, the dichotomies for 'Go' tasks are: 'Go' | 'RT Go' $\leftrightarrow$ 'Anti Go' | 'Anti RT Go' and 'Go' | 'Anti Go' $\leftrightarrow$ 'RT Go' | 'Anti RT Go'. Here, we train a classifier on 'Go' | 'RT Go' and test on 'Anti Go' | 'Anti RT Go', and vice versa. Repeating the process for the second dichotomy, we train on 'Go' | 'Anti Go' and test on 'RT Go' | 'Anti RT Go' and vice versa. To train and test decoders we use sklearn.svm.LinearSVC Python package. We set the maximum iterations to 5000 and otherwise keep all other parameters as default. For models trained on 'All Tasks' we repeat this process for all task groups and then average across all test decoder scores to obtain our reported results. For models trained with 'Heldout Tasks', we repeat each task group calculation four times, each time using representations obtained from a model that has one of the four tasks in the group held out of training. In this way we capture the degree of structure that results when each individual task is held out of training. Decoder scores are averaged to obtain the reported results. Error bars show +/- one standard deviation across model seeds.

Language Production Training

Language Production Network Training

Our language production framework is inspired by classic sequence to sequence modeling using RNNs [48]. We treat each time step in the sequence as a token position in the target sentence that we wish to decode. Our ProductionRNN is a GRU with 64 hidden units using ReLU non-linearities. 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. Importantly, the hidden state of the ProductionRNN is initialized by concatenating the time average and maximum sensorimotor activity of an *sbertNet-tuned* trained to perform all tasks and passing that through weights $\text{Linear}_{\text{sm}}$. This means that for an instructed model, training the ProductionRNN can proceed in a completely self-supervised manner. The linguistic instruction used to drive the initializing sensorimotor activity is in turn used as the target set of tokens for the ProductionRNN outputs. We also use dropout with a probability of 0.1 to prevent overfitting. The first input to the ProductionRNN is always a special start-of-sentence token (<SOS>) and the decoder runs until an end-of-sentence token (<EOS>) is decoded or until input reaches a length of 30 tokens. Suppose $w_{1,k} \dots w_{\mathcal{T},k} \in Instruct_k^i$ 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 transformers embed instructions as $Embed()$ (see Language Modules above). The decoded token at the τth position, $\hat{w}_{\tau,k}$, is then given by

$$
\begin{aligned}
h_T^{sm} &= \text{SensorimotorRNN}(X^i, Embed(w_{1,k} \dots w_{\mathcal{T},k})) && h_T^{sm} \in \mathbb{R}^{T \times 128} \\
sm_out &= \text{dropout}((\text{mean}_T(h_T^{sm}), \text{max}_T(h_T^{sm})) && sm_out \in \mathbb{R}^{256} \\
h_0^{\text{decoder}} &= \text{relu}(\text{Linear}_{\text{sm}}(sm_out)) && h_0^{\text{decoder}} \in \mathbb{R}^{64} \\
h_\tau^{\text{decoder}} &= \text{ProductionRNN}(\hat{w}_{1,k} \dots \hat{w}_{\tau-1,k}; h_0^{\text{decoder}}), && h_\tau^{\text{decoder}} \in \mathbb{R}^{64} \\
p_{\hat{w}_{\tau,k}} &= \text{softmax}(\text{Linear}_{\text{words}}(h_{\tau,k}^{\text{decoder}})) && p_{\hat{w}_{\tau,k}} \in \mathbb{R}^{|vocab|}, \\
\hat{w}_{\tau,k} &= \text{argmax}(p_{\hat{w}_{\tau,k}})
\end{aligned}
$$

The model parameters $\Theta^{\text{production}} = \{\text{Linear}_{\text{sm}}, \text{Linear}_{\text{words}}, \text{ProductionRNN}\}$ are trained using cross-entropy loss between the $p_{\hat{w}_{\tau,i}}$ and the instruction token $w_{\tau,k}$ provided to the SensorimotorRNN as input. We train for 80 epochs of 1000 batches with 32 trials each with task type randomly interleaved. We use an Adam optimizer with an initial learning rate of $lr = 0.001$ and a decay rate of $\gamma = 0.97$. We also use 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, $teacher_forcing_ratio = 0.5 * \frac{80-\text{epoch}}{80}$.

Supervised Embedding Vector Training Using Motor Feedback

We wish to determine whether or not networks can communicate task information gained during training on motor feedback, in the absence of linguistic instructions. To accomplish this we start with *sbertNet-tuned* models that have been trained to perform all tasks. For a given task, i, we seek to optimize a set of task identifying vectors $E^i \in \mathbb{R}^{20}$ such that when they are input to the network it will perform the task in question. Crucially, we freeze all model weights $\Theta = \{\text{SensorimotorRNN}, \text{Linear}_{\text{out}}\}$ and only update E^i according to the standard supervised loss on the motor output that comes from inputting E^i as task identifying information. For notional clarity GRU dependence on the previous hidden state h_{t-1} has been made implicit in the following equations.

$$
\begin{aligned}
\hat{y}^i &= \sigma(\text{Linear}_{\text{out}}(\text{SensorimotorRNN}(X^i, E^i))) \\
L &= mMSE(y, \hat{y})
\end{aligned}
$$

We use a set of 128 embedding vectors for each task and optimize this set in parallel as a single batch. Again using an Adam optimizer with an initial learning rate of $lr = 0.08$ which we decay by $\gamma = 0.99$ each epoch. An epoch lasts for 500 example trials and we train for a minimum of 20 epochs or until we reach a threshold performance of 95%.

Producing Task Instructions

To produce tasks instructions we simply use the set E^i as task identifying information in the input of the SensorimotorRNN and use the ProductionRNN to output instructions based on the sensorimotor activity driven by E^i. We repeat this process for each of the 5 initialization of networks trained on all tasks, resulting in 5 distinct language production networks, and 5 distinct sets of learned embedding vectors. Reported results are averaged over these 5 networks. For the confusion matrix (Fig. 5c), 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. 5d) for each network initialization is computed by testing each of the 4 possible partner networks and averaging over these results. Note that in the process of producing these instructions no explicit linguistic information has been provided to the network. It simply leverages its knowledge of the relationship between sensorimotor mappings and language to produce a good description of the task.

Methods Referneces

[45] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *CoRR*, abs/1508.05326, 2015.

[46] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[47] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[48] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[49] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.