Multitasking Recurrent Networks Utilize Compositional Strategies for Control of Movement

John Lazzari[1,2] and Shreya Saxena[1,2*]

[1]Center for Neurocomputation and Machine Intelligence, Wu Tsai Institute, Yale University, New Haven, CT, USA.
[2]Department of Biomedical Engineering, Yale University, New Haven, CT, USA.

*Corresponding author(s). E-mail(s): shreya.saxena@yale.edu;

The brain and body comprise a complex control system that can flexibly perform a diverse range of movements. Despite the high-dimensionality of the musculoskeletal system, both humans and other species are able to quickly adapt their existing repertoire of actions to novel settings. A strategy likely employed by the brain to accomplish such a feat is known as compositionality, or the ability to combine learned computational primitives to perform novel tasks. Previous works have demonstrated that recurrent neural networks (RNNs) are a useful tool to probe compositionality during diverse cognitive tasks. However, the attractor-based computations required for cognition are largely distinct from those required for the generation of movement, and it is unclear whether compositional structure extends to RNNs producing complex movements. To address this question, we train a multitasking RNN in feedback with a musculoskeletal arm model to perform ten distinct types of movements at various speeds and directions, using visual and proprioceptive feedback. The trained network expresses two complementary forms of composition: an algebraic organization that groups tasks by kinematic and rotational structure to enable the flexible creation of novel tasks, and a sequential strategy that stitches learned extension and retraction motifs to produce new compound movements. Across tasks, population activity occupied a shared, low-dimensional manifold, whereas activity across task epochs resides in orthogonal subspaces, indicating a principled separation of computations. Additionally, fixed-point and dynamical-similarity analyses reveal reuse of dynamical motifs across kinematically aligned tasks, linking geometry to mechanism. Finally, we demonstrate rapid transfer to held-out movements via simple input weight updates, as well as the generation of target trajectories from composite rule inputs, without altering recurrent dynamics, highlighting a biologically plausible route to within-manifold generalization. Our framework sheds light on how the brain might flexibly perform a diverse range of movements through the use of shared low-dimensional manifolds and compositional representations.

1 Introduction

In both cognitive and motor domains, animals can perform related but unfamiliar tasks with relatively little training. How does the brain achieve this remarkable flexibility? One widely proposed explanation is that cognitive and motor flexibility relies on the reuse and combination of previously learned computational primitives when appropriate—a strategy known as compositionality. For instance, a professional dancer can learn and perform new movements far more quickly than a novice, likely because they already possess foundational "motifs" that can be quickly recombined in novel ways. Moreover, they can retrieve and deploy the necessary patterns for executing a wide variety of movements—unlike modern control systems and deep learning models which largely focus on narrow settings.

There have been several recent studies experimentally exploring the neural basis of compositionality in cognitive settings [1–7]. These have subsequently inspired the exploration of compositional

representations in RNNs through the lens of multi-tasking or continual learning [8–12]. It has been shown that recurrent neural networks (RNNs) trained to perform multiple cognitive tasks develop modular computational units that are uniquely composed across tasks [8, 9]. However, this level of insight is yet to be seen from models performing various movement patterns, and it is unclear if the same principles apply in a continuous pattern generating setting without discrete motifs, such as those during working memory, decision making, or contextual inference tasks.

Indeed, compositional strategies during movement have been examined from multiple perspectives [13–15]. Popular topics regarding motor compositionality include hierarchical control and the use of muscle synergies to form complex movements. However, research in this regard has focused primarily on the capability of the nervous system to employ such strategies, not on motor cortical representations while stitching together primitive computations for the generation of diverse movements.

To model the motor cortex controlling the body, the joint use of RNNs and musculoskeletal models has been key for a diverse range of model organisms such as hydra, flies, rodents, and macaques[16–21]. Such embodied systems can reveal important structures in the networks trained to drive them. Although these frameworks have been used to study the correspondence between trained networks and recorded neural trajectories, they have largely not been leveraged for understanding multi-tasking and compositionality. Notable studies considering embodied control during multitasking are [18, 22], where rat skeletal models were trained to perform multiple tasks using reinforcement learning. The work of [22] in particular was the first to analyze the representation of multitasking networks performing movement related tasks. However, these works largely focus on representational similarities across diverse behaviors, as opposed to the underlying compositional strategies used by the network across tasks and how they are realized. For example, compositionality can be algebraic in nature, such that the neural representations of tasks are shifted in state space in accordance with a consistent and structured mapping, with different modes defining primitive components of the underlying computation [8]. Additionally, particularly in motor settings, kinematic motifs may be flexibly combined in novel ways to perform complex movements, which we term sequencing [23]. The ability of network models to form such representations strictly through multi-tasking has yet to be studied.

While experimental evidence for compositional representations in the motor cortex are lacking, there has been recent evidence suggesting the use of shared computations across tasks in primates [24–27]. Such reuse has been demonstrated through shared low-dimensional manifolds across diverse wrist-based motor tasks in macaques [24], consistent with the manifold hypothesis that the coordinated activity of neurons at the population level, not single-neuron tuning, is the primary computation towards movement generation [28, 29]. Through this lens, network modes *define* the building blocks of a computation, and a shared manifold allows for *reuse* of such modes across tasks. At the same time, studies have shown that the motor cortex uses orthogonal subspaces to isolate computations, particularly between preparatory and movement epochs [30–32]. Do networks trained to perform a variety of motor tasks adopt a similar organizational structure to share and isolate computations? If so, does this solution lend itself to compositional representations of movement?

Here, we develop a closed-loop multitasking RNN that controls a musculoskeletal arm to perform ten distinct types of movements, in order to examine whether a compositional structure emerges in the network through multitask training itself. The tasks include extensions and retractions of various movement patterns such as straight reaches, curved reaches, and more complex patterns. Compositional representations can be built from these tasks both algebraically, by grouping kinematic and rotational similarities in a structured manner, and sequentially, through the ability to stitch kinematic motifs together. We find that low-dimensional, shared subspaces are consistently utilized during the movement phase of all tasks, while computations across epochs are organized in orthogonal subspaces. The network also reflects dynamical similarities across aligned tasks, demonstrating reuse of dynamical structures. We then show that tasks are represented algebraically in network state space, and demonstrate that a task can be performed using motifs learned during other tasks through specific combinations of rule inputs. Additionally, we show that the kinematic motifs learned during training can be sequenced together without explicitly learning this relationship. Finally, we show the network is able to transfer its learned motifs to perform novel movements by only training the weights corresponding to a new static rule input. Overall, our framework provides a foundation for understanding the network structure and motifs that underlie motor compositionality.

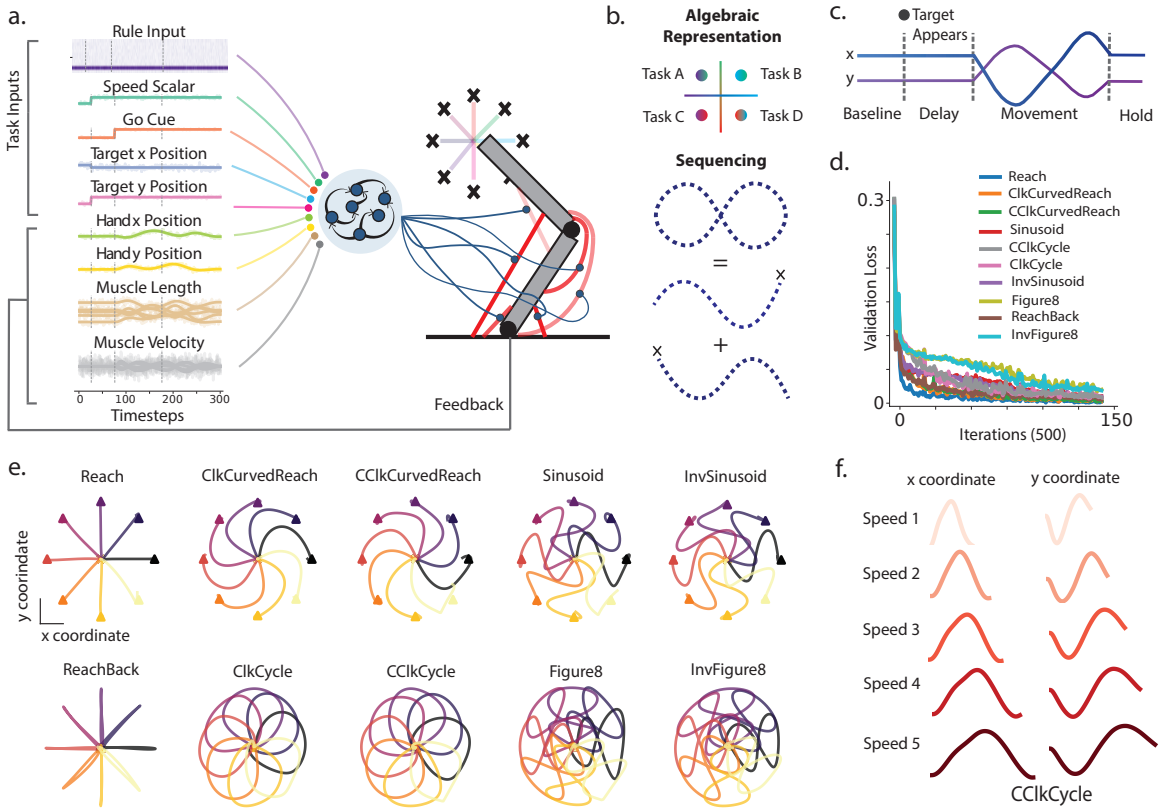

Fig. 1 Task design and training performance. (a) We train an RNN in feedback with a musculoskeletal model in order to perform ten distinct kinematic trajectories in eight different directions at three different speeds. Feedback from the arm is provided though muscle lengths, velocities, and visual feedback of the hand position. Other task-related variables are fed into the network such as the rule input, speed scalar, go cue, and target position. (b) Task design represents distinct kinematics containing composable elements. (top) Tasks can be constructed in an algebraic manner in network state space. (bottom) Individual extension and retraction motifs may be compartmentalized and sequenced in order to produce complex movements such as a figure eight. (c) Diagram of task structure. Task begins in a stabilization baseline period, where only the rule input appears, followed by a delay period in which condition-specific inputs appear, and finally movement occurs with a subsequent hold epoch. (d) Validation loss for interpolated conditions across all tasks. (e) Kinematic trajectories of the trained network for each task in each direction condition. (f) Example task kinematics for different speed conditions, with certain (not all ten) validation speeds included.

2 Results

2.1 Successful multi-task kinematics achieved through training

We design a suite of tasks analogous to the center-out paradigm commonly performed by primates in experimental settings (Fig. 1b) (see Supplemental for task details). The tasks share various computations, which may allow the network to repurpose learned motifs in unique ways. For example, the Figure8 task is composed of two sinusoidal movements, with the extension portion having the same kinematics as the Sinusoid task (Fig. 1b, bottom). Note that despite the kinematic similarities between these "subset" pairs (given that Sinusoid is a subset of the Figure8), they are distinguished by a one-hot contextual input representing individual tasks (i.e. the rule input). Additionally, the curved reaches, cycles, sinusoids, and figure eights each pose two distinct tasks, with the movement beginning either in a clockwise or counter-clockwise fashion. If a compositional representation forms in the network, the underlying kinematic patterns and rotational properties may be represented in a structured manner to construct each movement (Fig. 1b, top). There are five tasks with unique kinematics that only extend outwards, which we denote as extension tasks. Similarly, there are five tasks that extend outward and retract to the center using the same kinematic pattern, which we denote extension-retraction tasks. Each task contains a baseline stabilization epoch, followed by a delay epoch where condition specific inputs are active, then the movement epoch and subsequent hold epoch for stopping (Fig. 1c).

We trained an RNN using supervised learning to perform ten different motor tasks, with eight directions performed at three different speeds per task, for a total of 24 conditions (Fig. 1a). For the

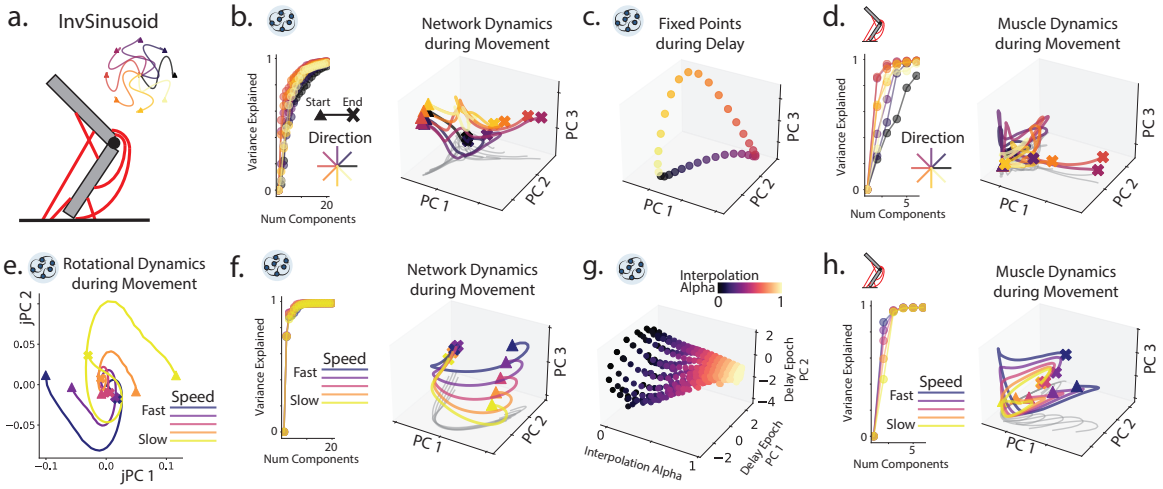

Fig. 2 Within-task dynamics of trained network. (a) Example task, in this case an inverse sinusoid (InvSinusoid). (b) (left) Variance explained ratio for each direction condition along the PCs captured across conditions. (right) Network trajectories along the top three PCs found across conditions. (c) Fixed point structure of network during the delay epoch is a ring representing each direction. (d) Same structure as (b) in the muscle space. (e) jPCA computed across speed conditions shows that the network utilizes rotational dynamics. (f) (left) Variance explained ratios for PCs captured across speed conditions. (right) Network trajectories projected onto the top three PCs shows speed conditions offset the same trajectories. (g) Fixed point interpolation between inputs during the delay to the middle of movement demonstrates large changes in fixed point structure between epochs. (h) Same structure as (f) in the muscle space.

arm musculoskeletal model, we use Motornet [16], a Python package for developing musculoskeletal models with differentiable muscles. The particular limb model we used is derived from [33], which implements a biologically plausible lumped-muscle model for 2D planar movements. The arm contains six muscles and two joints, and provides proprioceptive feedback to the network in the form of muscle lengths and velocities, and visual feedback of the hand position. Note that while the limb model we use here is not as complex as those shown in previous embodiment studies [17, 18, 22], we are able to replicate experimental results regarding both within and across-task computational strategies in both the network and muscle space, reflecting the accuracy of our approach. The simplicity of the arm model and task structure additionally presents a tractable framework for studying network representations and identifying motifs. Training resulted in accurate kinematic trajectories for all movements and conditions (Fig. 1e,f). The tasks were interleaved during training with equal probabilities, and the validation loss dropped consistently for each task, with Figure8 and InvFigure8 taking the longest to train and ending with the highest validation loss (Fig. 1d).

2.2 Within-task network dynamics are consistent with previous studies

We begin by examining how each individual task is solved by the network. Figure 2 shows an example of a single task, InvSinusoid, though other tasks exhibit similar structure (Supplemental Fig. 3). In Fig. 2b, we see that the network trajectories across reaches in different directions are consistent in orientation and shape but differ in offset or magnitude. This likely reflects the arm's geometry, suggesting that different conditions require slightly varied solutions. In Fig. 2 (c), we observe a ring of fixed points during the delay epoch, representing the initial condition for each movement [34, 35]. Interestingly, the fixed point structure changes drastically between epochs, as seen during interpolation between delay and movement inputs in Figure 2 (g). This suggests distinct dynamical computations across epochs, supported by the observation that delay-epoch PCs explain progressively less variance of movement-epoch fixed points during interpolation. In the muscle space (Fig. 2d), we observe less organization than in the network space, with some directions needing more dimensions to explain the same variance, along with heavily overlapping state space trajectories. Across speed conditions (Fig. 2f,h), network dynamics appear as translated versions of the same trajectory along a specific mode, while this structure is less present in muscle trajectories. This dichotomy is also observed in M1 and muscle recordings from non-human primates [36]. The RNN also engages rotational dynamics during reaches (Fig. 2e), a feature commonly observed across motor cortex tasks [37]. Overall, our model demonstrates a similar structural alignment for solving motor tasks as shown in previous experimental and modeling studies.

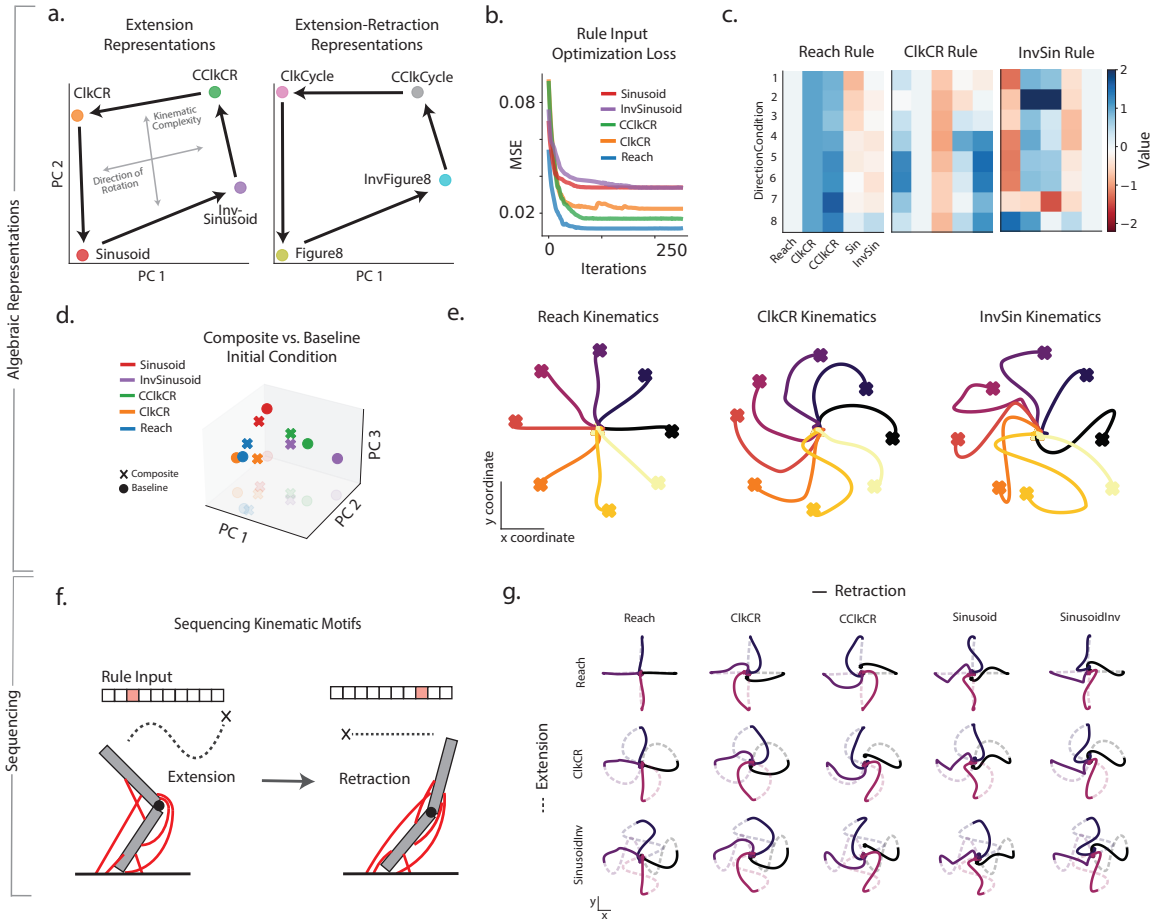

Fig. 3 Compositional representations. (a) Algebraic representation of tasks during the baseline epoch, when only the rule input is presented without condition specific information. PC 1 tends to capture rotational differences in tasks while PC 2 captures kinematic differences. (b) Optimizing for a combination of rule inputs necessary to perform a novel task. The Reach task was able to find the best combination of rule inputs for performance. All tasks are able to significantly minimize the loss. (c) Example rule inputs across eight direction conditions. Each rule input is unique across tasks and relatively consistent across conditions. (d) The initial conditions used by the learned rule inputs (x's) are similar in structure to the initial conditions used by the baseline model (circles). (e) Example kinematics from learned rule inputs shows that each task is performed properly. (f) Schematic of sequential compositionality and experimental setup. Rule inputs are switched during the middle of the movement epoch in order to perform unique kinematic motifs while extending and retracting. (g) All kinematic trajectories for sequential rule switching experiment designed in (f). Most tasks are able to perform well, while some fail to quickly reverse directions during retraction (e.g., CClkCR extension, ClkCR retraction). Despite this, many tasks and conditions are able to completely shift directions as well.

2.3 Distinct forms of compositionality emerges through task inputs

For RNNs performing many cognitive tasks, structured network representations emerge to organize tasks containing similar primitive motifs. In the current motor setting, where task outputs are defined by the continuous hand kinematics of an underlying musculoskeletal system, whether or not a compositional structure emerges is unclear. It is possible that the network develops an algebraic representation of tasks, as well as the ability to sequence various kinematic motifs. To test for algebraic representations, we use similar methods as [8]. First, we show a structured representation of tasks in state space using PCA on the averaged activity across conditions during the baseline epoch (Figure 3, a). We chose to visualize the baseline epoch representation since only the rule input (as opposed to other condition specific inputs) is considered, however other epochs have similar results (Supplemental Fig. 4). From Figure 3 (a), we can see a clear algebraic representation, with PC 1 distinguishing rotational components of the tasks and PC 2 capturing kinematic differences. This representation implies that tasks are built from primitive computations, and it may be possible to perform task A by applying specific combinations of rule inputs corresponding to different tasks that contain the necessary primitives [8]. To test this, we perform a rule input optimization to find the particular set of rule inputs for other tasks (not including task A) that allow for performing task A

well. The model is able to learn this rule input relatively well across tasks, with reaches and curved reaches performing the best (Fig. 3, b). Rule inputs are similar across conditions, and each task requires unique combinations of rule inputs (Fig. 3, c). These learned composite rule inputs bring the initial condition of the network at the end of the delay epoch relatively close to its initial condition using the original rule input (Fig. 3, d). Lastly, the resulting kinematic trajectories clearly represent the desired task (Fig. 3, e).

Another form of compositionality comes in the form of sequencing. In our task suite, the model learns both extension and retraction kinematic motifs, but only learns to combine them by using the same motif backwards (e.g., reach forward and reach back, cycles, figure eights). It is unclear if the model learned a modular representation for each kinematic motif such that any of them could be combined arbitrarily. To test this, we activate the rule input for a particular extension task, then once the movement is performed, switch to a rule input denoting a particular extension-retraction task to capture a different retraction motif, and see if both can be performed well (Fig. 3, f). The results of these experiments are shown in Figure 3 (g), with most tasks being able to perform distinct motifs during extension and retraction in an accurate manner. The network fails to perform well when the motif to be performed backwards counters the direction of the extension (e.g., ext: CClkCR and ret: SinusoidInv). Despite this limitation, the model is clearly able to combine unique extensions and retractions despite not being explicitly trained to do so, demonstrating that extension-retraction tasks are the result of reusing and composing learned kinematics in a modular fashion.

2.4 Network analysis shows a shared manifold across tasks yet distinct manifolds across epochs sub-tasks

The use of algebraic representations to organize tasks suggests the existence of shared network modes for similar tasks. Indeed, experimental evidence has shown that the motor cortex uses shared low-dimensional subspaces across distinct motor tasks [24]. While shared network subspaces do not directly imply compositional representations, it is likely that a compositional network utilizes similar structures as those found during experiments. This may suggest that such structures are likely necessary in order for compositional representations to emerge. To investigate, we computed the principal angles [38] between all pairs of tasks and between four different conditions within the same task. For across task comparisons, we tested the alignment between the top 12 principal components (PCs) found during the movement epoch—including trials from all direction conditions (Fig. 4a, left). For across condition comparisons, we similarly computed the top twelve PCs within each condition and tested their alignment for each task individually. Twelve PCs were chosen because they explained more than 95% variance during each task. Consistent with experimental findings in [24], we observe that the principal angles in the network space across tasks and conditions are generally low, except for a few modes. For comparison, we created a baseline control, where we generated a random m by N orthogonal basis—N being the number of units and m the dimension of the bases—and projected activity onto these random directions. To further quantify the correspondence of network modes across tasks and conditions, we calculated the variance explained ratios for the top twelve PCs (Fig.4a, right), where the ratio denotes the variance captured for task A by the task B PCs over the variance captured for task A by the task A PCs. The distribution of variance explained ratios is close to one for both task and condition comparisons, whereas the baseline is near zero. It is important to note that while these metrics assess the alignment between subspaces, neither of them require that the *same* PCs (e.g., PC 1 of two tasks) are aligned across tasks. We additionally explore modularity in this network and find the use of distinct populations across tasks (Supplemental Fig. 1).

Next, we test if the similarity across tasks and conditions found in the network space are due to analogous structures in the muscle output. We see that this is not the case in Figure 4 (b), as the percentage of similar modes across tasks and conditions in the muscle space is much lower than that of the network. In this setting, the number of similar modes is determined as the angles below the $P < 0.001$ baseline threshold, and the distribution of all possible task-condition comparisons are evaluated. Note that muscle activity overall is low-dimensional compared to network activity since there are far fewer muscles, however this metric is assessed relative to the baseline which accounts for the dimensionality of both systems. Furthermore, we compared the similarity between network and muscle latents across tasks using canonical correlation analysis (CCA), and find that muscle latents become uncorrelated more rapidly than network latents, in-line with the results in [24] (Fig. 4e). From this analysis, muscle activity appears to be relatively more complex than the network responses that generate them.

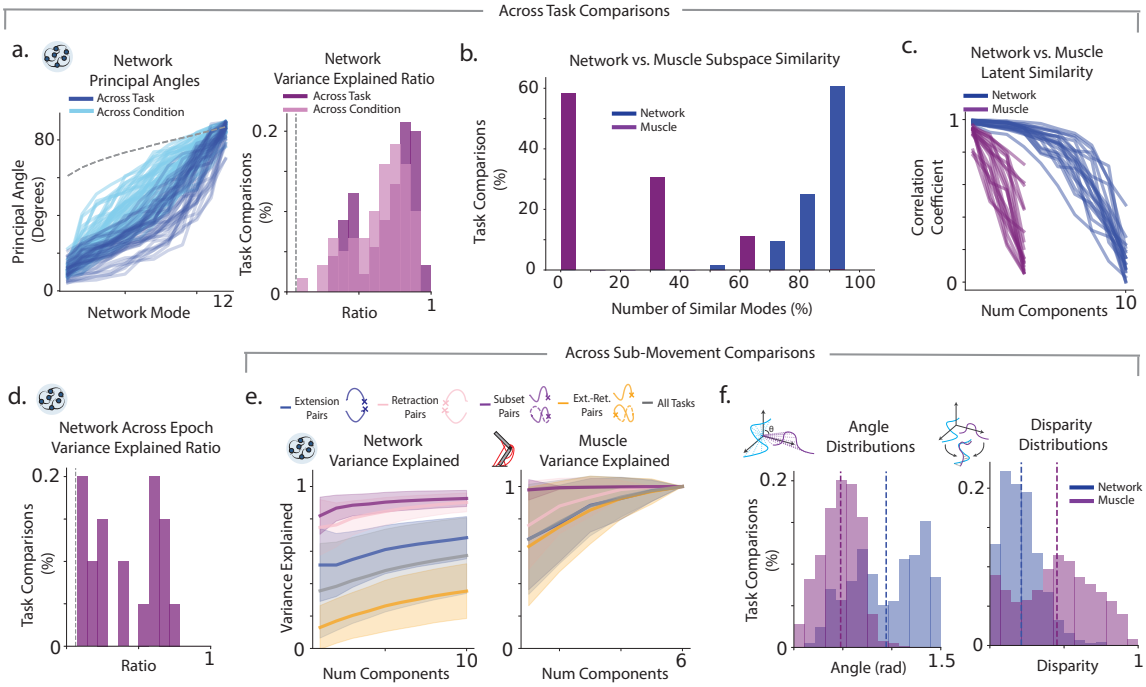

Fig. 4 Shared low dimensional representation across tasks and conditions. (a) (left) Principal angles for the top 12 PCs (explaining over 95% of the variance) for all tasks shows that many network modes are aligned. Across condition angles are similarly aligned relative to the baseline. (right) Ratio of variance explained for network activity projected onto the top 12 PCs of a different task or condition compared to its own. (b) Variance explained ratio for specific task pairs as the number of components increases. All movements except extensions and retractions are well aligned. Shading is standard deviation across direction conditions. (c) Variance explained ratio within task and across epochs. Modes are less aligned across epochs in comparison to cross-task modes. (d) Same as (a) for the muscle activity. (e) CCA on network and muscle activity across all task pairs shows that muscle activity becomes less similar in fewer modes. (f) Distributions of the angular distance and disparity between network trajectories and muscle trajectories across all task pairs. Angular distance has a wider distribution in the network space, and the opposite is true for the disparity distributions.

Despite the use of a shared low-dimensional manifold across tasks in the network space, it is unclear how sub-tasks within each task are organized. Examples of sub-tasks include epoch-specific computations, such as preparation and movement, and sub-movements, such as extensions and retractions. In Figure 4 (c) we show that the variance explained ratio *between different epochs within the same task* rapidly approach orthogonality in the network, in-line with experimental findings [31] (Fig. 4c). We next explore the subspace structure across sub-movements (Fig. 4, b (left)). We test the alignment between specific pairs of movements by plotting the variance explained ratio as the number of available components grows. The movement pairs include subsets, which have the same kinematics in distinct tasks (e.g., Reach and extension portion of ReachBack), any extension pair, any retraction pair, and extensions paired with retractions. We see that subset pairs are nearly perfectly aligned, suggesting much of the computation is reused in this setting despite differences in the rule input. Retraction and extension pairs are also relatively aligned, suggesting both categories individually reuse a shared subspace. However, the extension subspace is nearly orthogonal to the retraction subspace, as denoted by the low ratios between extension-retraction pairs. While such differences are present in the muscle space as well, each sub-movement is generally more aligned (Figure 4, b (right)). Further information regarding task alignment and example trajectories are shown in Supplemental Figure 2. Therefore, despite the use of a globally shared manifold across tasks, there are intrinsic orthogonal structures across sub-movements and epochs, primarily in the network space. In fact, by comparing the angular distance and disparity across all sub-movements in the high-dimensional muscle and network space, we can see that the angle distribution is widest in the network space and the disparity distribution is widest in the muscle space (Fig. 4, f). This suggests differing means of computation across sub-tasks, where the network primarily relies on the modes in which latent activity unfolds, and muscles primarily rely on differences in their time-dependent activation.

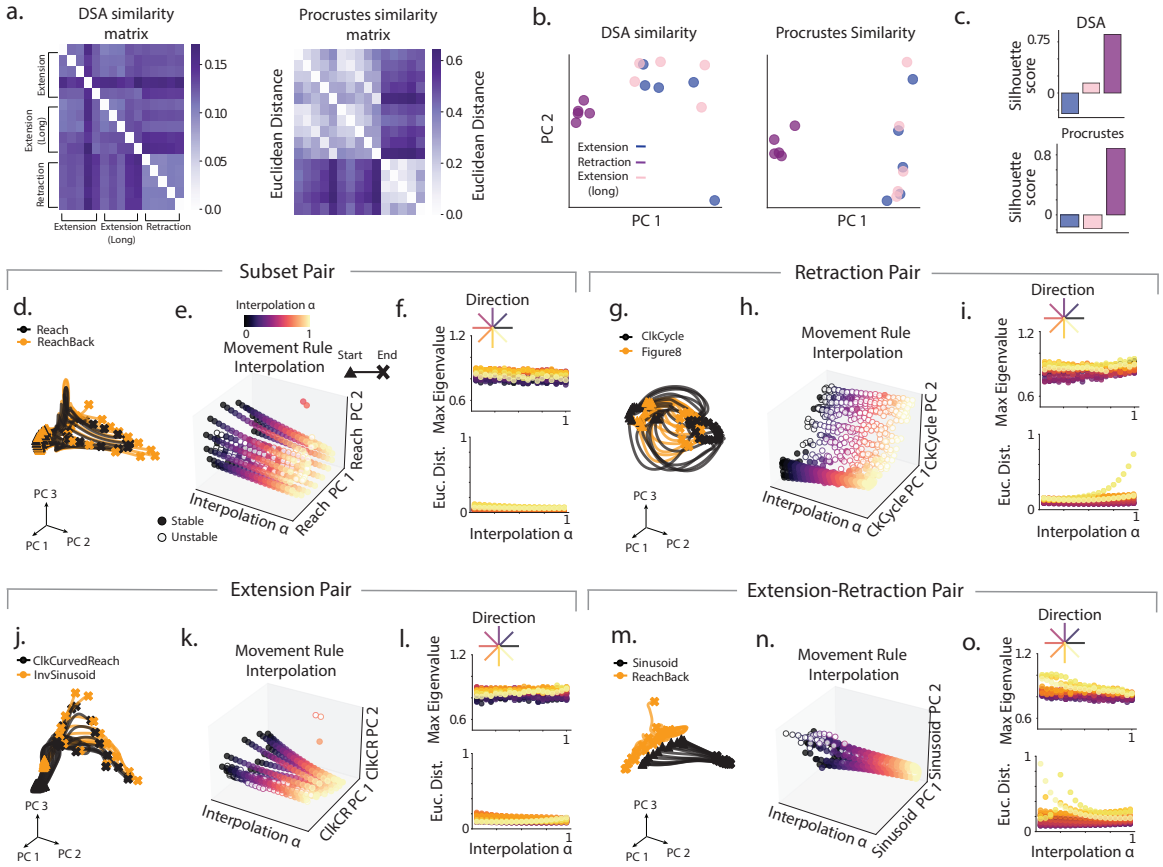

Fig. 5 Shifts in dynamical structure across tasks through input interpolation. (a) Procrustes and DSA dissimilarity matrices. Each row of the dissimilarity matrix is a vector containing the euclidean distance from the aligned trajectory (Procrustes) or vector field (DSA) of one task to all others. (b) Clustering of the 15x15 dissimilarity matrix using PCA. Retraction and extension tasks appear to be in separate clusters. (c) Quantification of cluster strength using averaged silhouette scores for each individual cluster. Both extension tasks (blue and pink) are close to zero due to their overlap. (d,e,f) Example task pair from subset category. (d) Trajectories projected onto PCs captured from the Reach task across all conditions shows strong overlap. (e) Fixed point structure of network for 20 values of α interpolating between the Reach ($\alpha = 0$) and ReachBack ($\alpha = 1$) tasks. Structure at beginning and end of interpolation is similar for this task pair. (f) (top) Maximum eigenvalue of network Jacobians found at each value of α across conditions. If a given α contains multiple fixed points, the maximum eigenvalue found across all computed Jacobians is plotted. (bottom) Euclidean distance between fixed points. The first fixed point is chosen as the one closest to the state trajectory at $\alpha = 0$. If a given α produces multiple fixed points, we choose the one that is closest to the chosen fixed point from the previous α value. (g,h,i) Same as (d,e,f) for a chosen retraction pair (ClkCycle, Figure8). (j,k,l) Same as (d,e,f) for a chosen extension pair (ClkCurvedReach, InvSinusoid). (m,n,o) Same as (d,e,f) for an extension-retraction pair (Sinusoid, ReachBack). The maximum eigenvalue distributions and fixed point distances seem to change the most across certain condition in this pair.

Note that the muscles recorded in [24] are of the hand and wrist, as opposed limb muscles, and performed a separate set of tasks not explored here. Despite experimental differences, our model is able to replicate findings regarding across-task subspace alignment and latent similarity, demonstrating the optimality of this strategy for a network performing many movements. In Supplemental Figures 5 and 6 we demonstrate that the subspace structures discussed in this section are consistent across hyperparameters.

2.5 Dynamical features are shared across movements in shared subspaces

While the network trajectory disparity across tasks is low relative to muscles, this does not fully characterize the dynamical structure of the network. It is possible for similar trajectories to have distinct underlying dynamics defining them [39]. However, if similar dynamics are used across pairs of tasks, this may demonstrate the reuse of learned computations. To investigate, we interpolate inputs using the methods developed in [9]—in our case including rule and feedback inputs during the middle of the movement epoch—and find fixed points along this trajectory to determine if large structural changes occur (Fig. 5). We project fixed points onto the across-condition task A PCs as

we interpolate towards the task B input. This will reveal changes in structure along the modes of task A as its PCs explain more or less of the variance for the fixed points. We perform this analysis across all direction conditions.

First, we globally quantify dynamical changes across movements (Fig. 5a,b,c). In Figure 5(a,b), we compute both shape and dynamical similarity analysis (DSA) [39] across a total 15 possible movements. In this case, we separately label extension movements (Extension), retraction movements (Retraction), and extension movements during tasks that contain retraction such as ReachBack (Extension (long)). We compute the 15x15 dissimilarity matrix containing the Euclidean distances found from alignment using Procrustes or Procrustes over vector fields (Fig. 5a), and visualize their clustering using PCA (Fig. 5b). For both DSA and Procrustes there is a clear separation between retraction and extension movements, with less clustering within each category (Fig. 5b). This is reflected in the silhouette scores across movement dissimilarity vectors, where retractions are strongly clustered and both extension categories are mixed due to their overlapping structure (Fig. 5c). Overall, these results demonstrate relatively stronger dynamical dissimilarities between extensions and retractions, primarily reflecting the differences in subspace structure observed in Figure 4.

We next focus on example task pairs, starting with the subset task pair (Fig. 5d,e,f). We chose the Reach and ReachBack tasks as an example. We can see that the trajectories for both tasks are overlapping on the Reach PCs (Fig. 5d). While this visualizes the similarities in their trajectory geometries, we can also qualitatively see that the fixed point structure is well maintained as we interpolate from the Reach to the ReachBack task (Fig. 5e). To quantify how dynamics change as we interpolate inputs, we linearize about a chosen fixed point for a particular α and plot the maximum eigenvalue of the corresponding network Jacobians (Fig. 5f) [9]. We similarly quantify the Euclidean distance between the chosen sets of fixed points along α (Fig. 5f) [9]. In both experiments, fixed points are chosen by first selecting the one closest to the state trajectory, then subsequently choosing the fixed point with the closest distance to the previous one. We can see that both the stability and distance between fixed points do not change much along α, denoted by their relatively constant values during interpolation. Between other task pairs such as Retraction (ClkCycle, Figure8, Fig. 5g,h,i) and Extension (ClkCurvedReach, InvSinusoid, Fig. 5j,k,l), we see similar trends in their dynamical similarity. The Extension-Retraction pair (Sinusoid, ReachBack, Fig. 5m,n,o) shows the most dynamical dissimilarity relative to the other task pairs. First, we can see that their trajectories are non-overlapping in the Sinusoid PC space (Fig. 5m). Additionally, the fixed point structure visualized in Figure 5 (n) appears to bifurcate more drastically at the beginning of interpolation for certain conditions. This is quantified further in Figure 5 (o), where certain conditions can be seen to more drastically change in both stability and fixed point distance during interpolation. These results also demonstrate the existence of condition-specific changes in dynamics across certain tasks, given that certain conditions are better maintained during interpolation than others, potentially caused by the arm's geometry. Clearly, many tasks share a similar dynamical structure, potentially allowing for compositional representations [9].

2.6 Transfer learning for novel movements

The utilization of compositional representations implies the ability to uniquely compose primitives to perform similar yet novel movements. Indeed, this ability is one of the hallmark features of compositionality, and a driving force behind its study. To demonstrate this ability, we first train the model on 8 of the 10 total tasks, here excluding CClkCurvedReach (CClkCR) and CClkCycle. After training, we freeze all weights in the model except for input weights of the two untrained rule inputs corresponding to the held-out tasks. We then test if the model is able to learn these tasks by simply reconfiguring learned motifs using the static rule inputs. Indeed, the model is able to perform well in this transfer learning setting, particularly for the CClkCR task (Fig. 6b,c). Performance in the CClkCycle task is lower than CClkCR, however this was true for the baseline model as well (Fig. 6b).

Next, we analyze the motifs utilized in order to solve both tasks. In Figure 6(d), we test how the model performs when the subset of tasks used during pre-training is limited. If the model does not require learned primitives from the pre-trained tasks, then restricting the pre-training to fewer subsets should have minimal impact on transfer learning. We test on six different subsets of tasks (none of which include the CClkCR and CClkCycle tasks for transfer learning): all available tasks (all), only sinusoidal tasks including figure eights (sin), only the Reach and ReachBack tasks (reach), all tasks except the sinusoidal and figure eight tasks (nosin), all tasks except ClkCR and ClkCycle (nocr), and only the ClkCR and ClkCycle tasks (cr). We see that pre-training setups that include

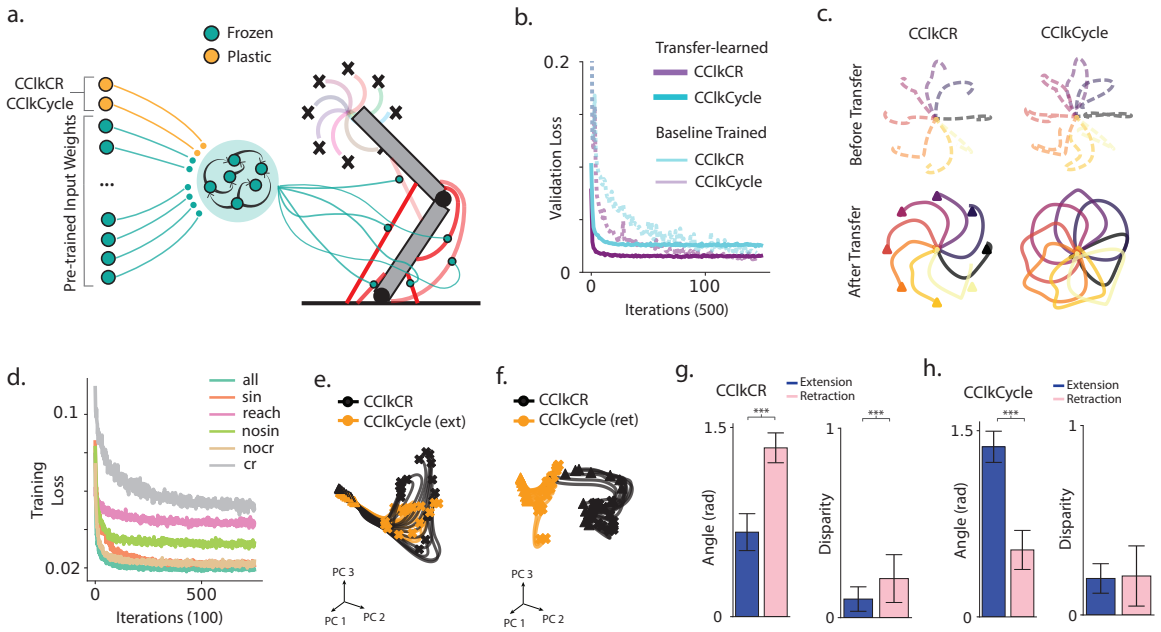

Fig. 6 Transfer of a pretrained model towards novel tasks. (a) Task setup—the model is pretrained on all tasks except for CClkCurvedReach (CClkCR) and CClkCycle, and all but the input weights for the rule inputs for these two tasks are frozen in the transfer setting. (b) Validation loss during transfer learning on the held-out tasks. Dashed lines denote baseline model performance on same tasks. (c) Example kinematics for both tasks before (top) and after (bottom) transfer learning. (d) Networks pretrained on fewer task subsets were not able to perform the held-out transfer learning tasks as well as those containing all possible task subsets. Training MSE loss shown during transfer learning of each pre-trained model. (e) Network trajectories for CClkCR and the extension portion of the CClkCycle task projected onto the top three CClkCR PCs. (f) Same as (e) but for the retraction portion of the CClkCycle task. (g) Angular distance and disparity metrics from the CClkCR task to all other extension and retraction tasks. CClkCR trajectories are better aligned with the extension tasks as opposed to retraction tasks in both metrics ($p < 0.001$, angular distance; $p < 0.001$, disparity; Mann-Whitney U test). Error bars denote standard deviation across pairs. (h) Same as (g) but for the retraction portion of the CClkCycle task. As expected, this task is closely aligned with other retraction tasks ($p < 0.001$, angular distance; Mann-Whitney U test).

more primitives (e.g., nocr, sin) generally perform better during transfer learning, while those that don't perform worse. However, including all available tasks during pre-training still results in the best performance for the held-out tasks. This demonstrates the necessity of learned primitives from each task during transfer learning.

Next, we observe that the structure learned during pre-training is reused during transfer learning. In Figure 6 (e), we visually see that the CClkCR task and the extension portion of the CClkCycle task have overlapping latent trajectories, suggesting strong alignment in their solutions. This is in contrast to the CClkCR task and the retraction portion of the CClkCycle task shown in Figure 6 (f), which reflects the orthogonality between extension and retraction movements discussed previously. In Figure 6(g,h), we further quantify how the transfer learned tasks are aligned with previously learned subspace structures. We utilize the angular distance metric to test state space alignment, and disparity using Procrustes to test geometric alignment. We see that the CClkCR task is primarily aligned with other extension movements as opposed to retractions, demonstrating its utilization of a previously learned extension subspace for solving this novel task (Fig. 6g). For the retraction portion of the CClkCycle task, we instead see greater alignment between other retraction movements, as expected (Fig. 6h). Overall, these results demonstrate the ability of our model to learn novel tasks using previously encountered motifs, a hallmark feature of compositional representations.

3 Discussion

In the complex and unpredictable natural world, the ability to flexibly adapt, learn, and deploy a wide array of skilled movements is vital. To better understand the underlying motifs that allow networks to reconfigure computations in complex control settings, we design a model and task structure amenable to systematic analyses of shared computations across contexts. RNNs have been increasingly utilized

as tools for hypothesis testing in the field of motor control, and have subsequently provided insightful and biologically accurate predictions [29, 40]. More recently, these models have been extended to include biomechanical feedback [17, 18], an integral component of the motor system's functioning [41]. Indeed, we find that our biologically plausible model shares various solution-based similarities as experimental and experiment-adjacent modeling studies. Our task structure closely represents laboratory settings in which monkeys are trained to produce movements in a center-out fashion, allowing us to draw close comparisons with experimental findings. Additionally, our suite of kinematic motifs are simple and well-defined, allowing for interpretable results in terms of task representations and shared computations.

In recent years, there has been a shift from studying the tuning of individual units to examining population-level dynamics during the control of movement. The analysis of population-level structures during movement control has subsequently led to the discovery of low-dimensional manifolds and a focus on the most prominent neural modes for interpreting task-relevant computations [42]. Moreover, such manifolds are not exclusive to primate movement, having been discovered brain-wide in various species, including monkeys [43–46], rats [47, 48], Aplysia [49], and zebrafish [50]. We find shared network subspaces across tasks and conditions in our model. This is in line with experimental findings [24], suggesting that the overarching commonalities between distinct motor tasks are represented using the same neural modes, allowing for their reuse as building blocks for movement generation. We then find orthogonal structures across epochs and across sub-movements within tasks, such as extensions and retractions, suggesting that the network either reuses or orthogonalizes subspaces depending on their computational similarity.

The computational principles underlying the network and muscles appear to differ based on their subspace structures and temporal alignment. Both systems utilize low-dimensional manifolds across tasks, however the muscle modes are relatively closer to the random orthogonal basis across conditions in comparison to the network modes. This likely reflects constraints imposed by the arm's geometry, which minimally affects network subspace alignment. Such metrics however do not fully characterize the computational differences between both systems, given that any available modes captured throughout the full movement can be aligned. Metrics such as angular distance and Procrustes on the high-dimensional network and muscle trajectories, compared across all sub-movements, further reveal that the network more widely distributes active modes while the muscles more widely distribute the time-dependent similarity between trajectories. This suggests that the shared low-dimensional structures present in the network do not solely reflect similar structures in muscles, but largely result in unique temporal patterns of muscle activity.

In cognitive settings such as decision making, the underlying attractor dynamics necessary to perform tasks are well defined. For example, continuous attractors are consistently implemented by recurrent networks tasked with accumulating evidence to a decision threshold [8, 45]. Conversely, during motor control, the underlying dynamics necessary to produce particular movements are less obvious. To assess dynamical similarity in the motor setting, we provide a fixed point analysis and DSA during movement. We find that movements performed using aligned network subspaces demonstrate stronger dynamical similarities in comparison to unaligned movements, both in terms of changes in fixed point structure and disparity over vector fields. This demonstrates the reuse of dynamical structures across similar tasks within the network, in line with [9].

The use of common subspaces and dynamical structures hints towards the emergence compositional representations. We found that RNNs implement an algebraic representation of tasks, distinguishing their kinematic structure and rotational properties. This form of compositionality is commonly seen in cognitive settings where motifs are more clearly defined [8]. However, the existence of such motifs is less obvious within the continuous spectrum of movement. Provided such motifs, task representations can potentially be combined in unique ways to construct distinct tasks, which we found was true. Additionally, RNNs are capable of performing unique extensions and retractions in sequence without explicitly being trained to do so. This suggests that the network learns to compartmentalize extension and retraction motifs in order to sequence them, as opposed to forming a wholly unique representation for each individual extension-retraction task. This may be due to the reuse of motifs learned during extension tasks, which is suggested in Figure 3 and Figure 5.

One significant advantage of compositional representations is the ability to flexibly adapt in novel settings. Multi-tasking models in cognitive settings have indeed shown the ability to increase the learning speed of new tasks after pre-training on similar tasks [51]. They are also able to learn new tasks solely using previously learned motifs [9]. Our network demonstrates the same capabilities through novel combinations of previously learned motifs for learning new movements. The model will

additionally reuses the same subspace structures learned during pre-training. The results from Figure 6 are thus aligned with [52], where primates were able to quickly learn within-manifold perturbations of a center-out reach task. Changing the network dynamics to learn out of manifold tasks should instead occur on a longer timescale and necessitate a reorganization of network connectivity [27].

One limitation of this model is the relatively low number of muscles in the arm, along with its planar two-dimensional structure. It is possible that with greater redundancy—i.e., more muscles and a higher-dimensional task space—there may be a more complex relationship between network and muscle modes. Additionally, it is hypothesized that the simple low dimensional manifolds typically found during lab experiments—as well as in our setting—may reflect the simplicity of the tasks themselves. The addition of full body skeletal or musculoskeletal models capable of performing a wide array of naturalistic behaviors may enhance our understanding of the (approximately) true underlying motor manifold, which may be non-linear and higher dimensional in nature. Hierarchical structures in the motor system likely also play a vital role in understanding the compositional nature underlying motor control. Framing the motor cortex as a high-level controller may abstract the representations found in this paper to instead depend on the combination and interaction between low-level controllers. Despite the simplicity of our current setup, our framework was able to clearly identify both shared and isolated computations between movements and epochs in both the muscle and network space, in addition to compositional representations. Our framework can thus provide insight regarding the compositional structure used by networks to achieve flexible control of movement, and can be adapted in the future to include more complex musculoskeletal models and a wider variety of tasks reflecting less stereotyped paradigms.

References

[1] Voitov, I., Mrsic-Flogel, T.D.: Cortical feedback loops bind distributed representations of working memory. Nature **608**(7922), 381–389 (2022)

[2] Tafazoli, S., Bouchacourt, F.M., Ardalan, A., Markov, N.T., Uchimura, M., Mattar, M.G., Daw, N.D., Buschman, T.J.: Building compositional tasks with shared neural subspaces. bioRxiv (2024)

[3] Langdon, C., Genkin, M., Engel, T.A.: A unifying perspective on neural manifolds and circuits for cognition. Nature Reviews Neuroscience **24**(6), 363–377 (2023)

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

[5] Yang, G.R., Cole, M.W., Rajan, K.: How to study the neural mechanisms of multiple tasks. Current opinion in behavioral sciences **29**, 134–143 (2019)

[6] Frankland, S.M., Greene, J.D.: Concepts and compositionality: in search of the brain's language of thought. Annual review of psychology **71**(1), 273–303 (2020)

[7] Willett, F.R., Deo, D.R., Avansino, D.T., Rezaii, P., Hochberg, L.R., Henderson, J.M., Shenoy, K.V.: Hand knob area of premotor cortex represents the whole body in a compositional way. Cell **181**(2), 396–409 (2020)

[8] 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. Nature neuroscience **22**(2), 297–306 (2019)

[9] Driscoll, L.N., Shenoy, K., Sussillo, D.: Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. Nature Neuroscience **27**(7), 1349–1363 (2024)

[10] Duncker, L., Driscoll, L., Shenoy, K.V., Sahani, M., Sussillo, D.: Organizing recurrent network dynamics by task-computation to enable continual learning. Advances in neural information processing systems **33**, 14387–14397 (2020)

[11] Dubreuil, A., Valente, A., Beiran, M., Mastrogiuseppe, F., Ostojic, S.: The role of population

structure in computations through neural dynamics. Nature neuroscience **25**(6), 783–794 (2022)

[12] Beiran, M., Dubreuil, A., Valente, A., Mastrogiuseppe, F., Ostojic, S.: Shaping dynamics with multiple populations in low-rank recurrent networks. Neural computation **33**(6), 1572–1615 (2021)

[13] d'Avella, A., Giese, M., Ivanenko, Y.P., Schack, T., Flash, T.: Modularity in motor control: from muscle synergies to cognitive action representation. Frontiers Media SA (2015)

[14] Todorov, E.: Compositionality of optimal control laws. Advances in neural information processing systems **22** (2009)

[15] Merel, J., Botvinick, M., Wayne, G.: Hierarchical motor control in mammals and machines. Nature communications **10**(1), 5489 (2019)

[16] Codol, O., Michaels, J.A., Kashefi, M., Pruszynski, J.A., Gribble, P.L.: Motornet, a python toolbox for controlling differentiable biomechanical effectors with artificial neural networks. Elife **12**, 88591 (2024)

[17] Almani, M.N., Lazzari, J., Chacon, A., Saxena, S.: μ sim: A goal-driven framework for elucidating the neural control of movement through musculoskeletal modeling. bioRxiv, 2024–02 (2024)

[18] Aldarondo, D., Merel, J., Marshall, J.D., Hasenclever, L., Klibaite, U., Gellis, A., Tassa, Y., Wayne, G., Botvinick, M., Ölveczky, B.P.: A virtual rodent predicts the structure of neural activity across behaviours. Nature **632**(8025), 594–602 (2024)

[19] Wang, H., Swore, J., Sharma, S., Szymanski, J.R., Yuste, R., Daniel, T.L., Regnier, M., Bosma, M.M., Fairhall, A.L.: A complete biomechanical model of hydra contractile behaviors, from neural drive to muscle to movement. Proceedings of the National Academy of Sciences **120**(11), 2210439120 (2023)

[20] Lobato-Rios, V., Ramalingasetty, S.T., Özdil, P.G., Arreguit, J., Ijspeert, A.J., Ramdya, P.: Neuromechfly, a neuromechanical model of adult drosophila melanogaster. Nature Methods **19**(5), 620–627 (2022)

[21] Chiappa, A.S., Tano, P., Patel, N., Ingster, A., Pouget, A., Mathis, A.: Acquiring musculoskeletal skills with curriculum-based reinforcement learning. Neuron **112**(23), 3969–3983 (2024)

[22] Merel, J., Aldarondo, D., Marshall, J., Tassa, Y., Wayne, G., Olveczky, B.: Deep neuroethology of a virtual rodent. In: International Conference on Learning Representations (2019)

[23] Logiaco, L., Abbott, L., Escola, S.: Thalamic control of cortical dynamics in a model of flexible motor sequencing. Cell reports **35**(9) (2021)

[24] Gallego, J.A., Perich, M.G., Naufel, S.N., Ethier, C., Solla, S.A., Miller, L.E.: Cortical population activity within a preserved neural manifold underlies multiple motor behaviors. Nature communications **9**(1), 4233 (2018)

[25] Gallego, J.A., Perich, M.G., Miller, L.E., Solla, S.A.: Neural manifolds for the control of movement. Neuron **94**(5), 978–984 (2017)

[26] Golub, M.D., Sadtler, P.T., Oby, E.R., Quick, K.M., Ryu, S.I., Tyler-Kabara, E.C., Batista, A.P., Chase, S.M., Yu, B.M.: Learning by neural reassociation. Nature neuroscience **21**(4), 607–616 (2018)

[27] Oby, E.R., Degenhart, A.D., Grigsby, E.M., Motiwala, A., McClain, N.T., Marino, P.J., Yu, B.M., Batista, A.P.: Dynamical constraints on neural population activity. Nature Neuroscience, 1–11 (2025)

[28] Shenoy, K.V., Sahani, M., Churchland, M.M.: Cortical control of arm movements: a dynamical systems perspective. Annual review of neuroscience **36**(1), 337–359 (2013)

[29] Vyas, S., Golub, M.D., Sussillo, D., Shenoy, K.V.: Computation through neural population dynamics. Annual review of neuroscience **43**(1), 249–275 (2020)

[30] Kaufman, M.T., Churchland, M.M., Ryu, S.I., Shenoy, K.V.: Cortical activity in the null space: permitting preparation without movement. Nature neuroscience **17**(3), 440–448 (2014)

[31] Elsayed, G.F., Lara, A.H., Kaufman, M.T., Churchland, M.M., Cunningham, J.P.: Reorganization between preparatory and movement population responses in motor cortex. Nature communications **7**(1), 13239 (2016)

[32] Churchland, M.M., Shenoy, K.V.: Preparatory activity and the expansive null-space. Nature Reviews Neuroscience **25**(4), 213–236 (2024)

[33] Nijhof, E.-J., Kouwenhoven, E.: Simulation of multijoint arm movements. In: Biomechanics and Neural Control of Posture and Movement, pp. 363–372. Springer, ??? (2000)

[34] Inagaki, H.K., Chen, S., Daie, K., Finkelstein, A., Fontolan, L., Romani, S., Svoboda, K.: Neural algorithms and circuits for motor planning. Annual review of neuroscience **45**(1), 249–271 (2022)

[35] Sun, X., O'Shea, D.J., Golub, M.D., Trautmann, E.M., Vyas, S., Ryu, S.I., Shenoy, K.V.: Cortical preparatory activity indexes learned motor memories. Nature **602**(7896), 274–279 (2022)

[36] Saxena, S., Russo, A.A., Cunningham, J., Churchland, M.M.: Motor cortex activity across movement speeds is predicted by network-level strategies for generating muscle activity. Elife **11**, 67620 (2022)

[37] Churchland, M.M., Cunningham, J.P., Kaufman, M.T., Foster, J.D., Nuyujukian, P., Ryu, S.I., Shenoy, K.V.: Neural population dynamics during reaching. Nature **487**(7405), 51–56 (2012)

[38] Bjorck, A., Golub, G.H.: Numerical methods for computing angles between linear subspaces. Mathematics of computation **27**(123), 579–594 (1973)

[39] Ostrow, M., Eisen, A., Kozachkov, L., Fiete, I.: Beyond geometry: Comparing the temporal structure of computation in neural circuits with dynamical similarity analysis. Advances in Neural Information Processing Systems **36**, 33824–33837 (2023)

[40] Sussillo, D., Churchland, M.M., Kaufman, M.T., Shenoy, K.V.: A neural network that finds a naturalistic solution for the production of muscle activity. Nature neuroscience **18**(7), 1025–1033 (2015)

[41] Todorov, E., Jordan, M.I.: Optimal feedback control as a theory of motor coordination. Nature neuroscience **5**(11), 1226–1235 (2002)

[42] Cunningham, J.P., Yu, B.M.: Dimensionality reduction for large-scale neural recordings. Nature neuroscience **17**(11), 1500–1509 (2014)

[43] Kobak, D., Brendel, W., Constantinidis, C., Feierstein, C.E., Kepecs, A., Mainen, Z.F., Qi, X.-L., Romo, R., Uchida, N., Machens, C.K.: Demixed principal component analysis of neural population data. elife **5**, 10989 (2016)

[44] Machens, C.K., Romo, R., Brody, C.D.: Functional, but not anatomical, separation of "what" and "when" in prefrontal cortex. Journal of Neuroscience **30**(1), 350–360 (2010)

[45] Mante, V., Sussillo, D., Shenoy, K.V., Newsome, W.T.: Context-dependent computation by recurrent dynamics in prefrontal cortex. nature **503**(7474), 78–84 (2013)

[46] Markowitz, D.A., Curtis, C.E., Pesaran, B.: Multiple component networks support working memory in prefrontal cortex. Proceedings of the National Academy of Sciences **112**(35), 11084–11089 (2015)

[47] Raposo, D., Kaufman, M.T., Churchland, A.K.: A category-free neural population supports

evolving demands during decision-making. Nature neuroscience **17**(12), 1784–1792 (2014)

[48] Chapin, J.K., Nicolelis, M.A.: Principal component analysis of neuronal ensemble activity reveals multidimensional somatosensory representations. Journal of neuroscience methods **94**(1), 121–140 (1999)

[49] Bruno, A.M., Frost, W.N., Humphries, M.D.: Modular deconstruction reveals the dynamical and physical building blocks of a locomotion motor program. Neuron **86**(1), 304–318 (2015)

[50] Ahrens, M.B., Li, J.M., Orger, M.B., Robson, D.N., Schier, A.F., Engert, F., Portugues, R.: Brain-wide neuronal dynamics during motor adaptation in zebrafish. Nature **485**(7399), 471–477 (2012)

[51] Goudar, V., Peysakhovich, B., Freedman, D.J., Buffalo, E.A., Wang, X.-J.: Schema formation in a neural population subspace underlies learning-to-learn in flexible sensorimotor problem-solving. Nature Neuroscience **26**(5), 879–890 (2023)

[52] Sadtler, P.T., Quick, K.M., Golub, M.D., Chase, S.M., Ryu, S.I., Tyler-Kabara, E.C., Yu, B.M., Batista, A.P.: Neural constraints on learning. Nature **512**(7515), 423–426 (2014)

[53] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[54] Sussillo, D., Barak, O.: Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. Neural computation **25**(3), 626–649 (2013)

[55] Brunton, S.L., Brunton, B.W., Proctor, J.L., Kaiser, E., Kutz, J.N.: Chaos as an intermittently forced linear system. Nature communications **8**(1), 19 (2017)

4 Methods

4.1 Network and arm structure

We train standard RNNs in feedback with a musculoskeletal model to perform ten unique movements designed to elucidate a compositional structure in the network activity (Fig. 1a,b). The form of the network is described below:

$$\begin{aligned} \tau\dot{x} &= -x + W_h h + W_{\text{in}} u + b_h + \sqrt{2\tau\delta^2}\epsilon \\ h &= \sigma(x) \end{aligned} \tag{1}$$

Here, τ represents the network time constant, set to $\tau = 20$, W_h, b_h, and W_{in} represent the hidden weights, hidden bias, and input weights respectively, $\epsilon \sim \mathcal{N}(0,1)$, δ is a noise scalar set to $\delta = 0.1$, and σ is the network activation.

We discretize 1 in order to train the network using conventional deep learning methods; the form of the discretized network is given as

$$\begin{aligned} x_t &= (1-\alpha)x_{t-1} + \alpha(W_h h_{t-1} + W_{\text{in}} u_t + b_h) + \sqrt{2\tau\delta^2}\epsilon \\ h_t &= \sigma_{\text{rec}}(x_t), \end{aligned} \tag{2}$$

The term $\alpha = \Delta t/\tau$, where $\Delta t = 20$. We utilize a readout layer that transforms network outputs to muscle stimulations in the range $(0, 1)$

$$z_t = \sigma_{\text{out}}(W_{\text{out}} h_t + b_{\text{out}})$$

In this case σ_{out} is the sigmoid function $\sigma_{\text{out}}(x) = \frac{1}{1+e^{-x}}$. The output layer formulation is consistent across all hyperparameter settings due to the requirements of the arm.

For the musculoskeletal model we use Motornet [16], an open-source framework for designing musculoskeletal models with differentiable muscles in Python. This allowed us to directly backpropagate

through the muscles and train the network using supervised learning. The model itself contains six Hill-type muscles and two joints, capturing horizontal movements at the shoulder level. Each muscle represents a lumped representation of synergistic muscle groups in a primate arm, with a total of 19 muscles considered [33].

4.2 Task formulation and training

For each task the arm starts in the same initial joint state—representing a center-out setup paradigm (Fig. 1a)—then is instructed to move in one of eight directions at one of three different speeds. For validation, we save the model that has the best performance on held out, interpolated conditions including 32 directions and 10 speeds. The RNN receives proprioceptive feedback in the form of muscle lengths and velocities, and visual feedback of the end effector position (Fig. 1a). Other task-related inputs include a speed scalar, coordinates of the target position, go cue, and one-hot rule input. Four epochs characterize a trial, including a baseline stability epoch, a delay epoch where condition inputs become active, movement epoch, and lastly a holding epoch (Fig. 1c). For each task, the length of the baseline T_{baseline} and hold T_{hold} epochs are $T_{\text{baseline}} = 25$, $T_{\text{hold}} = 25$. For the delay epoch, the length is randomly selected according to $T_{\text{delay}} \sim \{50, 75, 100\}$. The length of the movement epoch during training is determined by the chosen speed condition, and ranges from $T_{\text{move}} \sim \{50, 100, 150\}$ for extension only tasks and $T_{\text{move}} \sim \{100, 200, 300\}$ for tasks containing extension and retraction.

The kinematics of the tasks themselves are designed to contain compositional elements (Fig. 1b). The task suite is as follows: reach (Reach), clockwise and counter clockwise curved reach (ClkCurvedReach, CClkCurvedReach), sinusoid (Sinusoid) and inverse sinusoid (InvSinusoid), reach outward and backward (ReachBack), clockwise and counterclockwise cycles (ClkCycle, CClkCycle), and lastly figure eight and inverse figure eight (Figure8, InvFigure8). While each task in its entirety represents a unique kinematic motif, there are various compositional elements. For example, the extension kinematics of the ClkCycle task are the same as the ClkCurvedReach task, certain tasks only involve extension, while some involve retraction, and some tasks are simply rotated versions of another (e.g. ClkCycle, CClkCycle).

The network is trained on five separate losses, given as

$$
\begin{aligned}
L_1 &= \|g(y_{i,t}) - \hat{y_{i,t}}\|_2 \\
L_2 &= \frac{1}{N} \sum_i \|f(h, x)\|_1 \\
L_3 &= \frac{1}{N} \sum_i \|W_h\|_1 \\
L_4 &= \frac{1}{N} \sum_i \|g(y)\|_1 \\
L_5 &= \frac{1}{CT} \sum_c \int_0^T \left\| \frac{\partial W_h f(c,t)}{\partial x(c,t)} \right\|^2 \\
L &= \sum_i \alpha_i L_i
\end{aligned}
$$

The function g is the mapping from muscle activity to hand kinematics (musculoskeletal dynamics), the function f is the forward pass of the RNN, W_h are the recurrent weights of the network, and x is the network preactivation. The first loss L_1 is the mean squared error between the hand position and the desired trajectory position. Note that the model does not receive this desired position as feedback. Losses L_2, L_3, and L_4 represent l1 penalties on the network rates, weights, and muscle activities respectively. Lastly, we utilize the simple dynamics loss presented in [40] for L_5. Our total loss is the sum of each individual loss scaled by a weighting factor α_i. For the base model shown in the paper we use $\alpha_1 = 10^{-3}, \alpha_2 = 10^{-3}, \alpha_3 = 10^{-3}, \alpha_4 = 10^{-2}, \alpha_5 = 10^{-3}$. We train the model for $75,000$ iterations using a batch size of 32, and 256 units for the base model. The optimizer used is Adam [53], with a learning rate of 10^{-3}.

For loss L_1, we generate desired kinematics for each task. The methods for doing so are presented below:

- **Reach**: Generate a line from the initial hand position $z_0 \in \mathbb{R}^2$ to the target position $\hat{p} \in \mathbb{R}^2$: $\hat{t} = z_0(1-\alpha) + \alpha\hat{p}$, $\alpha \in [0, 1]$. The target position is a random orientation on the unit circle $p \in \mathbb{R}^2$, chosen from eight equally spaced bins, scaled and shifted by $\beta, \delta \in \mathbb{R}$ respectively to be centered at the hand position, $\hat{p} = \beta p + \delta$.
- **ClkCurvedReach**: Generate a scaled and shifted curved trajectory $\hat{t}(x_i) = \beta(\cos(x_i), \sin(x_i)) + \delta, \quad \beta, \delta \in \mathbb{R}$. In this case $x_i = \pi(1-\alpha_i), \quad \alpha_i = \frac{i}{T}, i = 0, 1, 2, ..., T$ where T is the chosen number of timepoints.
- **CClkCurvedReach**: Same as ClkCurvedReach except $x_i = \pi(1-\alpha_i) + 2\pi\alpha_i$.
- **Sinusoid**: Generate a scaled and shifted curved trajectory $\hat{t}(x_i) = \beta(x_i, \sin(y_i)) + \delta, \quad \beta, \delta \in \mathbb{R}$, where $x_i = \frac{i}{T}, \quad y_i = 2\pi\alpha_i, \quad \alpha_i = \frac{i}{T}, i = 0, 1, 2, ..., T$.
- **InvSinusoid**: Same as Sinusoid except $\hat{t}(x_i) = \beta(x_i, -\sin(y_i)) + \delta$
- **ReachBack**: We combine the trajectory used in the Reach task, $\hat{t}_1$, with a separate trajectory from the target position back to the center: $\hat{t}_2 = \hat{p}(1-\alpha) + \alpha z_0$. The total trajectory is defined as $\hat{t} = [\hat{t}_1, \hat{t}_2]$.
- **ClkCycle**: We combine the trajectory used in the ClkCurvedReach task, $\hat{t}_1$, with a separate trajectory $\hat{t}_2$ in the same direction circling back to the initial condition using a distinct x_i: $x_i = -\pi\alpha_i, \quad \alpha_i = 1 - \frac{i}{T}, i = 0, 1, 2, ..., T$. The total trajectory is defined as $\hat{t} = [\hat{t}_1, \hat{t}_2]$.
- **CClkCycle**: Same method as ClkCycle, using the CClkCurvedReach trajectory as $\hat{t}_1$, except the x_i used for $\hat{t}_2$ is given as $x_i = \pi\alpha_i, \quad \alpha_i = 1 - \frac{i}{T}, i = 0, 1, 2, ..., T$.
- **Figure8**: We combine the trajectory used in the Sinusoid task, $\hat{t}_1$, with a separate trajectory $\hat{t}_2 = \beta(x_i, -\sin(y_i)) + \delta, \quad \beta, \delta \in \mathbb{R}$. Here, $x_i = 1 - \frac{i}{T}, y_i = 2\pi\alpha_i, \quad \alpha_i = 1 - \frac{i}{T}, i = 0, 1, 2, ..., T$. The total trajectory is defined as $\hat{t} = [\hat{t}_1, \hat{t}_2]$.
- **InvFigure8**: Same method as Figure8, using the InvSinusoid trajectory as $\hat{t}_1$.

4.3 Manifold Analysis

In order to compose previously learned motifs across tasks, the network may utilize a shared space in which the same network modes are reconfigured to perform similar movements in distinct settings. To test whether or not a shared low dimensional manifold exists across tasks, we utilize the methods presented in [24, 38] to compute the principal angles between network modes across tasks. Briefly, given X_A and X_B, the top m principal components (PCs) for tasks A and B respectively that explain above 95% of task variance, we compute the principal angles between these modes by computing C via $X_A^T X_B = P_A C P_B^T$ using singular value decomposition, where X_A and X_B are the n by m matrices containing the top m PCs for tasks A and B, and P_A and P_B are the m by m matrices defining the new manifold directions that minimize the principal angles. The essence of this method is to successively find modes that align with each other for both tasks. The matrix C contains the ranked cosines of the principal angles $C = \text{diag}(\cos(\theta_1), \cos(\theta_2), ..., \cos(\theta_m))$. In addition to principal angles, we also compute the ratio of variance explained between the PCs across and within tasks. The form of this ratio is simply $\text{var}(X_B h_A)/\text{var}(X_A h_A)$. The baseline for both metrics is a random orthogonal m by N basis, built using a QR decomposition, where m is the number of basis vectors chosen for computing principle angles (in this case 12 for the network and three for the muscles). To define the number of similar modes across task and condition comparisons in Figure 4 (b), we take the 0.1 percentile of 5000 comparisons of randomly generated bases to create a $P < 0.001$ threshold, below which the observed principle angles in the network and muscles are considered significant.

In Section 2.4, we group task pairs based on their similarities and dissimilarities then compute their angular distance to test their alignment. The angular distance is time-averaged as such

$$\theta = \arccos\left(\frac{1}{T}\sum_t \frac{x_t^T y_t}{||x_t||||y_t||}\right)$$

When grouping kinematic pairs, we separate extension-retraction tasks into two halves to compare both phases of the movement distinctly. The task pairs are as such:

- Subset: Tasks with different rule inputs but the same kinematics (e.g., Reach and the extension half of ReachBack).
- Extension: Any pair of extensions excluding subsets (e.g., ClkCR and Sinusoid)
- Retraction: Any pair of retractions, found from the second half of extension-retraction tasks (e.g. retraction phase of Figure8 and ClkCycle).

- Extension Retraction: Any extension paired with any retraction (e.g. Reach and retraction phase of Figure8).

4.4 Latent Analysis

In order to test the alignment of network and muscle latents, we utilize CCA and Procrustes analysis. CCA aims to find the modes such that the single-dimensional latent activity of two datasets A and B are maximally correlated. For CCA, we first compute the first ten PCs of task A and B separately, then project the activity found across conditions for these tasks onto their corresponding PCs to obtain the latent activity matrices L_A and L_B. CCA begins with a QR decomposition of the transposed latent activity matrices as such $L_A^T = Q_A R_A, L_B^T = Q_B R_B$. Singular value decomposition is then performed on the inner product matrix of Q_A and Q_B as such

$$Q_A^T Q_B = USV^T$$

The elements of S contains the sorted canonical correlations which we use to determine the similarity between latents across tasks. We use the same pairs discussed in Section 4.3.

For Procrustes, we do not perform PCA on network activity before testing alignment. Procrustes optimizes the following objective

$$d(X, Y) = \min_{C \in O(n)} ||X - CY||_2$$

In this case, X and Y are TxN dimensional time-series, and $O(n)$ is the orthogonal group $C^T C = I$. This metric will compare the similarity between the temporal patterns produced by each trajectory regardless of their orientation, offset, or scaling.

4.5 Composite Rule Inputs

In order to test the model's ability to combine primitive motifs to form novel movements, we tested whether or not motifs learned from other tasks can be uniquely combined in order to perform a separate task. Let $u^e \in \mathbb{R}^5$ denote the first five components of u for a particular condition. These components represent the rule inputs corresponding to the extension tasks. We optimize for a $\zeta \in \mathbb{R}^5$ element-wise scaled by u^e such that $||g(f(h, \zeta \odot u^e, u^{5:N_i}), \hat{y}||_2$ is minimized, where N_i is the input size. For each desired task $\hat{y}$, the rule input corresponding to that task in u^e is 0. We learn distinct rule inputs across conditions for better performance.

4.6 Sequencing Kinematic Motifs

To test if individual kinematic motifs can be sequenced, we switch the rule inputs corresponding to extension and retraction tasks at the middle of the movement epoch. Let u^{r_i} denote the input for task i and u^{r_j} denote the rule input for task j. The total input to the network is then $u = [u^{r_i}_{1:t_{\text{middle}}}, u^{r_j}_{t_{\text{middle}}:t_{\text{end}}}]$, where t_{middle} and t_{end} denote the middle and end of the movement epoch respectively. In this setting, u^{r_i} is always an extension task (e.g., Reach) and u^{r_j} is always an extension-retraction task (e.g., ReachBack).

4.7 Fixed Point Analysis

In order to assess dynamical similarities across task pairs, we analyze the network fixed point structures and local dynamics around the fixed points. We specifically interpolate between the rule inputs for different tasks and determine whether the fixed point structure is preserved [9]. We perform this separately for the delay and movement epochs of the motor task. To find fixed points, we follow methods used in [9, 54]. We optimize to find a set of hidden states $\{h_1^*, h_2^*, ...\}$ that minimize $F(h^*, u^*)$, where u^* is a fixed input along the interpolated trajectory $\alpha u_1 + (1 - \alpha)u_2$, and $F(h, u)$ is 1. In this case, u_1 and u_2 are the inputs for two different tasks, either at the end of the delay epoch or the middle of the movement epoch.

In addition to finding fixed points, we assess their local dynamics to determine if changes in stability occur as we interpolate. To do so, we linearize the network about the fixed point h^*:

$$f(h^* + \delta h^*, u^* + \delta u^*) \approx f(h^*, u^*) + \frac{\partial f}{\partial h}(h^*, u^*)\delta h + \frac{\partial f}{\partial u}(h^*, u^*)\delta u$$

By definition, $f(h^*, u^*) = 0$, and second order terms are approximately zero given that $||\delta h||^2 \approx 0$. Additionally, we keep our desired input u^* constant, therefore we can ignore any changes from δu. Thus, our Taylor expansion simplifies to

$$f(h^* + \delta h^*, u^* + \delta u^*) \approx \frac{\partial f}{\partial h}(h^*, u^*)\delta h$$

The eigenvalues of the linearized system's Jacobian, $\frac{\partial f}{\partial h}(h^*, u^*)\delta h$, tells us the local dynamics near fixed points. As we interpolate from one task to another, we find the largest eigenvalue of the computed Jacobian for each fixed point found during optimization to assess changes in stability. To choose the fixed points used for plotting, we first select the fixed point closest to the state trajectory at $\alpha = 0$. Afterwards, we choose the fixed point at α_{i+1} closest to the one previously selected for α_i. We chose this method to best capture the trajectory of a single fixed point as we interpolate from one task to another. In addition to viewing max eigenvalues, we additionally quantify change in the structure of fixed points by taking their euclidean distance, $d(h_t^*, h_{t+1}^*) = ||h_t^*, h_{t+1}^*||_2$, during interpolation. This metric uses the same method for selecting fixed points as previously mentioned.

4.8 Dynamical Similarity Analysis

In Section 2.5 we perform DSA to identify clusters of dynamical and geometrical similarity between tasks. Given two dynamical systems

$$\dot{x} = f(x, t) \qquad \dot{y} = g(y, t)$$

DSA samples data from each system, $X \in \mathbb{R}^{c \times k_1 \times t_1 \times n}$ and $Y \in \mathbb{R}^{c \times k_1 \times t_1 \times n}$, where c is the number of conditions, k is the number of trials per condition, t is the number of timepoints, and n is the number of units in the system. Then, the data is used to create delay-embedded Hankel tensors with lag of p: $H_x = \in \mathbb{R}^{c \times k_1 \times (t_1 - p - 1) \times np}$ [55]. The Hankel Alternative View of Koopman (HAVOK) approach is then used to build the necessary dynamic mode decomposition (DMD) matrices. This approach flattens all but the last dimensions of H and fits reduced rank regression models with rank r to the coordinates of the next time step $V_{x,r}^{'}$:

$$\begin{aligned} V_{x,r}^{'} &= A_x V_{x,r} \\ H_x^T &= U_x \Sigma_x V_x^T \\ V_{x,r} &= V_x[:, 1:r] \end{aligned} \tag{3}$$

The resulting DMD matrices are given by A_x and A_y. Lastly, Procrustes analysis is performed to test the alignment between the DMD matrices through a similarity transformation. The optimization problem posed by Procrustes analysis over vector fields is given as:

$$d(A_x, A_y) = \min_{C \in O(n)} ||A_x - C A_y C^{-1}||_2$$

In Figure 5, we perform this analysis on separate tasks to test their dynamical similarity. Instead of using the ten tasks in totality, we also split the extension and retraction portions of the extension-retraction tasks to more clearly test the learned dynamical structure across kinematic motifs. To test the strength of the clustering given by the labeled extension and retraction tasks, we take the silhouette score of each cluster, defined as

$$s(d_{i,\text{inter}}, d_{i,\text{intra}}) = 1 - \frac{d_{i,\text{intra}}}{d_{i,\text{inter}}}$$

Where $d_{i,\text{inter}}$ is the average distance between unit i and every unit in the nearest cluster, and $d_{i,\text{intra}}$ is the average distance between unit i and every unit in the same cluster. The average across all units i is used to score a cluster. A low score denotes weak clustering, while a score close to one denotes otherwise.

4.9 Transfer Learning

To test whether the network can reconfigure learned motifs to quickly learn new tasks, we train the network on all tasks except CClkCR and CClkCycle, then freeze the learned weights $W_h, W_{\text{out}}, W_{\text{in}}, b_h, b_{\text{out}}$ and only learn vectors u_{cr} and u_{cycle} within W_{in} to perform the held out tasks. We exclude both CClkCR and CClkCycle to ensure that the network does not learn any counter-clockwise curved reach motif during any task. To test whether the motifs learned during pretraining are necessary for performing the transfer learning tasks well, we train the model on stricter subsets of pre-trained tasks to determine if performance on the held out tasks are affected. The additional task subsets are as follows:

- sin: {Sinusoid, InvSinusoid, Figure8, InvFigure8}
- reach: {Reach, ReachBack}
- nosin: {Reach, ClkCR, ReachBack, ClkCycle}
- nocr: {Reach, Sinusoid, InvSinusoid, ReachBack, Figure8, InvFigure8}
- cr: {ClkCR, ClkCycle}

We additionally test whether learned subspace structures are reused during transfer learning. To do so, we compute the angular distance and disparity between CClkCR with all other extension and retraction movements, and similarly for CClkCycle.

4.10 Varying Hyperparameters

We trained networks with variable activation functions and hidden units to test if the subspace structures found using the baseline model are consistent across hyperparameters. The activation functions include Softplus

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

the ReLU function

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

and the Tanh function

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

We test models with 128, 256, and 512 hidden units for each activation function. Supplemental Figures 5 and 6 demonstrate the results of this hyperparamter search and demonstrate that shared low-dimensional subspaces across tasks are consistently used in different settings.

4.11 Task Variance Analysis

In Supplemental Figure 1, we perform a task variance analysis to determine if populations of neurons are differentially selective across tasks. We use the same task variance metric as [8], where the variance is computed across all conditions and timesteps for a particular task then averaged. The form of the task variance metric is given as

$$TV_i(A) = \langle [h_i(j,t) - \langle h_i(j^{'},t)_{j'} \rangle]^2 \rangle_{j,t}$$

where i denoted the current unit, A represents the task, j is the condition, and t is time. In our case, we compute this metric over all speed and direction conditions during the movement and delay epochs. Units that have a summed task variance across all tasks less than 10^{-3} were discarded for inactivity.

Given the task variance of each unit, we aim to find clusters of task selectivity in the population. To do so, we used k-means clustering on the task variances while ranging the number of clusters from 2 to 20. To choose the best clustering, we compute the silhouette score, averaging over all units, then choose the highest score.