Published in final edited form as:
Nat Neurosci. 2018 December ; 21(12): 1774–1783. doi:10.1038/s41593-018-0276-0.

Motor primitives in space and time via targeted gain modulation in cortical networks

Jake P. Stroud[1,@], **Mason A. Porter**[2,3,4], **Guillaume Hennequin**[5], and **Tim P. Vogels**[1]

[1]Centre for Neural Circuits and Behaviour, University of Oxford, Oxford, UK

[2]Department of Mathematics, University of California Los Angeles, Los Angeles, USA

[3]Mathematical Institute, University of Oxford, Oxford, UK

[4]CABDyN Complexity Centre, University of Oxford, Oxford, UK

[5]Computational and Biological Learning Lab, Department of Engineering, University of Cambridge, Cambridge, UK

Abstract

Motor cortex (M1) exhibits a rich repertoire of neuronal activities to support the generation of complex movements. Although recent neuronal-network models capture many qualitative aspects of M1 dynamics, they can generate only a few distinct movements. Additionally, it is unclear how M1 efficiently controls movements over a wide range of shapes and speeds. We demonstrate that modulation of neuronal input–output gains in recurrent neuronal-network models with fixed architecture can dramatically reorganize neuronal activity and thus downstream muscle outputs. Consistent with the observation of diffuse neuromodulatory projections to M1, a relatively small number of modulatory control units provide sufficient flexibility to adjust high-dimensional network activity using a simple reward-based learning rule. Furthermore, it is possible to assemble novel movements from previously learned primitives, and one can separately change movement speed while preserving movement shape. Our results provide a new perspective on the role of modulatory systems in controlling recurrent cortical activity.

[@]Questions or additional requests should be addressed to JPS (jake.stroud@cncb.ox.ac.uk).

Data availability
Data sharing is not applicable to this article as no datasets were generated or analysed for this study.

Author contributions
JPS, GH, and TPV conceived the study and developed the model. JPS performed simulations for Figs. 1–5 and Supplementary Figs. 1–4, and JPS and GH performed simulations for Figs. 6–8 and Supplementary Figs. 5–8. JPS analyzed the results, produced the figures, and wrote the first draft of the manuscript. All authors discussed and iterated on the analysis and its results, and all authors revised the final manuscript.

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

Code availability
We include sample MATLAB code a http://modeldb.yale.edu/246004.

Introduction

Motor cortex is one of the final cortical outputs to downstream spinal motoneurons [1], and it is fundamental for controlling voluntary movements [2, 3, 4]. During movement execution, primary motor cortex (M1) exhibits complex, multiphasic firing-rate transients that return to baseline after movement completion [4]. Recent studies have provided some understanding of how these complex, single-neuron patterns of activity relate to intended movements [4, 5, 6]. It has been insightful to view motor cortex as a dynamical system in which preparatory activity sets the initial condition for the system, whose subsequent dynamics drive the desired muscle activity [7, 8]. From this perspective, the complex firing-rate dynamics provide a flexible basis set for the generation of movements [9].

Several recurrent neuronal-network models have been developed to capture M1 activity during movement execution [10, 11]. These models rely on strong recurrent connectivity that is optimized for the neuronal dynamics to be qualitatively similar to M1 activity during movement execution. However, these models cannot explain how new movements can be constructed or how their static architecture allows variations in both output trajectories and speed.

A possible mechanism for effectively switching neuronal activity, and consequently downstream muscle activity to generate different movements (see Fig. 1a), is to adjust the intrinsic gain — that is, the input–output sensitivity — of each neuron so that they engage more (or less) actively in the recurrent neuronal dynamics [12, 13, 14, 15, 16, 17, 18]. Indeed, neuromodulation in M1 can cause such changes in neuronal responsiveness [19, 20], and gain modulation of both neurons in M1 [13] and spinal motoneurons [21, 22] has been linked experimentally to skill acquisition and optimization of muscular control.

In this paper, we study the effects of gain modulation in recurrent neuronal-network models of motor cortex. We show that individually modulating the gain of neurons in such models allows learning of a variety of target outputs on behaviourally relevant time scales through reward-based training. Motivated by diffuse neuromodulatory innervation of M1 [19, 23, 24], we find that coarse-grained control of neuronal gains achieves a similar performance to neuron-specific modulation. We demonstrate that we can combine previously learned modulatory gain patterns to accurately generate new desired movements. Therefore, gain patterns can act as motor primitives for quickly constructing novel movements [25, 26]. Finally, we show how to control the speed of an intended movement through gain modulation. We find that it is possible to learn gain patterns that affect either only the shape or only the speed of a movement, thus enabling efficient and independent movement control in space and time.

Results

Modelling gain modulation in recurrent neuronal networks

To understand how cortical networks can efficiently generate a large variety of outputs, we begin with an existing cortical circuit model [11]. We use recurrent networks, with $N = 2M$

neurons (with M excitatory and M inhibitory neurons), for which the neuronal activity vector $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))^{\mathrm{T}}$ evolves according to

$$\tau \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \boldsymbol{W} f(\boldsymbol{x}(t); \boldsymbol{g}), \quad (1)$$

where the single-neuron time constant is $\boldsymbol{\tau} = 200$ ms, and (unless we state otherwise) we generate the synaptic weight matrix $\boldsymbol{W}$ in line with [11] (i.e., we use 'stability-optimized circuits'). These networks consist of a set of sparse, strong excitatory weights that are balanced by fine-tuned inhibition (see Methods).

The gain function f, which governs the transformation of neuronal activity $\boldsymbol{x}$ into firing rates relative to a baseline rate r_0, is

$$f(x_i; g_i) = \begin{cases} r_0 \tanh\left(g_i x_i / r_0\right), & \text{if } x_i < 0, \\ \left(r_{\max} - r_0\right) \tanh\left(g_i x_i / \left(r_{\max} - r_0\right)\right), & \text{if } x_i \geq 0, \end{cases} \quad (2)$$

where the gain g_i is the slope of the function f at the baseline rate r_0 and thus controls the input–output sensitivity of neuron i [27]. In Eqn. (1), $f(\boldsymbol{x};\boldsymbol{g})$ denotes the element-wise application of the scalar function f to the neuronal activity vector $\boldsymbol{x}$. Unless we state otherwise, we use a baseline rate of $r_0 = 20$ Hz and a maximum firing rate of $r_{\max} = 100$ Hz, consistent with experimental observations [4, 28]. The gain function $f(\boldsymbol{x};\boldsymbol{g})$ describes the neuronal firing rates relative to the baseline steady-state r_0. Identical dynamics can also result from using a strictly positive gain function, combined with a tonic (i.e., static) external input (see Methods Section 1.2).

For appropriate initial conditions $\boldsymbol{x}(t = 0) = \boldsymbol{x}_0$ (see Methods Section 1.1), the neuronal dynamics given by Eqn. (1) exhibit naturalistic activity transients that resemble M1 recordings [4, 11], and the population activity is rich enough to enable the generation of complex movements through linear readouts [11]. We emulate neuromodulation in this model by directly controlling the input–output gain g_i of each neuron (see Figs. 1b,c).

Neuron-specific gain modulation

We find that increasing the gain of all neurons uniformly (i.e., $g_i = g$ in Eqn. (2)) increases both the frequency and amplitude of the neuronal firing rates (see Fig. 1c). One can understand these effects of uniform modulation by linearizing Eqn. (1) around $\boldsymbol{x} = \boldsymbol{0}$, yielding the linear ordinary differential equation $\tau \frac{dx}{dt} = (g\boldsymbol{W} - \boldsymbol{I})\boldsymbol{x}$ (where $\boldsymbol{I}$ is the identity matrix), and studying changes in the spectrum of the matrix $g\boldsymbol{W} - \boldsymbol{I}$; see Supplementary Math Note.

To allow more precise control of neuronal activity than through uniform modulation, we can independently adjust the gain of each neuron in what we call *neuron-specific modulation*. We obtain gain patterns that lead to the generation of target output activity using a reward-based node perturbation learning rule (see Methods Section 1.7). Our rule, which acts on the modulatory pathway of our model but is similar to proposed synaptic plasticity rules for reward-based learning [29, 30, 31, 32], uses a global scalar signal of recent performance to iteratively adjust each neuron’s gain while the initial condition $\boldsymbol{x}_0$ and the network architecture remain fixed.

Starting with a network and readout weights that produce an initial movement with all gains set to 1 (see the black curve in Fig. 1d), our learning rule yields a gain pattern that leads to the successful generation of a novel target movement after a few thousand training iterations (see Fig. 1d and Methods Section 1.10). Errors between the actual and desired outputs tend to decrease monotonically and eventually become negligible. Independent training sessions with the same target movement produce nonidentical but positively correlated gain patterns (see Fig. 1e and Supplementary Fig. 1c). Counterintuitively, the neuronal firing rates change only slightly, even though the network output is altered substantially (see Supplementary Fig. 1b). Once the target is learned, the same initial condition can produce either of two distinct network outputs, depending on the applied gain pattern (see Fig. 1f). The outputs are also similarly robust with respect to noisy initial conditions for each gain pattern (see Supplementary Fig. 1d).

We also compare the learning performance of gain modulation with alternative learning mechanisms. We train either the neuronal gains, the initial condition $\boldsymbol{x}_0$ of the neuronal activity, a rank-1 perturbation of the synaptic weight matrix, or the full synaptic weight matrix using back-propagation (see Methods Section 1.10). We find empirically for this task that training through gain modulation yields a similar learning performance as training the initial condition or the full synaptic weight matrix and that training through gain modulation performs substantially better than learning a rank-1 perturbation of the synaptic weight matrix (see Supplementary Fig. 1f).

Gain modulation in different models

Next, we examine whether learning through gain modulation is possible in alternative, commonly-used models of movement generation. Motor circuits that drive movements also engage in periods of movement preparation [5, 7, 33], suggesting a role for gain modulation in shaping circuit dynamics both during movement planning and during movement execution. We find that learning is also possible in a model in which we include gain modulation during movement planning. We simulate the preparatory period using a ramping input to the system [11] (see Methods Section 1.10), such that gain modulation now directly affects the neuronal activity at movement onset. We find that learning performance (i.e., error reduction) for the task that we showed in Fig. 1d is slightly poorer if we do employ a ramping input than if we do not. (Compare the red and blue curves in Fig. 2a.) This occurs because gain modulation during the preparatory phase changes the neuronal activity at movement onset, allowing it to leave the null space of the readout weights (which are fixed) and thus elicit premature muscle activity at movement onset.

We also construct a ‘chaotic’ variant of our model [34] (see Methods Sections 1.3 and 1.10) for the same task and train only the neuronal gains. We achieve similar learning performance compared to our original model that we used in Fig. 1d (compare the red and grey curves in

Fig. 2a), even though the neuronal firing rates are very different (compare the far left and far right panels in Fig. 2b). Finally, we also use an alternative learning rule to train the neuronal gains (see Eqns. (10) and (11)); in this rule, learning slows down as the decrease in error slows down (see Methods Section 1.8). We find that the error decreases at a faster rate than that in our original learning rule (see the purple curve in Fig. 2a.) This may occur because the standard deviation of the noise perturbation term in the alternative learning rule becomes smaller over training iterations as the error decreases.

Notably, in all of these examples, changes in neuronal responsiveness alone — for example, via inputs from neuromodulatory afferents — can cause dramatic changes in network outputs, thereby providing an efficient mechanism for rapid switching between movements, without requiring any changes in either synaptic architecture or the initial condition $\boldsymbol{x}_0$.

Coarse, group-based gain modulation

Individually modulating the gain of every neuron in motor cortex is likely unrealistic. In line with the existence of diffuse (i.e., not neuron-specific) neuromodulatory projections to M1 [19, 24, 23], we cluster neurons into groups so that we identically modulate units within a group. (See Fig. 3a and Methods Sections 1.9 and 1.10.) We find that such coarse-grained modulation gives similar performance to neuron-specific control for as few as 20 randomly-formed groups (see Methods Section 1.9) using our model from Fig. 1 consisting of 200 neurons (see Fig. 3b and Supplementary Fig. 2a). For a given number of groups, one can improve performance if, instead of grouping neurons randomly as above, we use a specialized clustering for each movement that is based on previous training sessions (see Fig. 3b, Supplementary Fig. 2a, and Methods Section 1.9). Importantly, there exist specialized groupings that perform similarly across multiple different movements (see Fig. 3c and Supplementary Figs. 2b,c). Such specialized groupings acquired from learning one set of movements also perform well on novel movements (see Supplementary Fig. 2d).

Notably, even with random groupings, network size hardly affects learning performance for a single readout (see Fig. 3d). The performance depends much more on the number of groups than on the number of neurons per group. When the task involves two or more readout units, larger networks do learn better, and achieving a good performance necessitates using a larger number of independently modulated groups (see Figs. 3e,f). Finally, smaller networks typically learn faster (see the bottom panel of Fig. 3e), but they ultimately exhibit poorer performance, demonstrating that there is a trade-off between network size, number of groups, and task complexity (i.e., the number of readout units).

Gain patterns can provide motor primitives for novel movements

In principle, it is possible to independently learn numerous gain patterns, supporting the possibility of a repertoire (which we call a ‘library’) of modulation states that a network can use, in combination, to produce a large variety of outputs. Generating new movements is much more efficient if it is possible to ‘intuit’ new gain patterns as combinations of previously acquired primitives [26, 15]. To test if this is possible in our model, we first approximate a novel target movement as a convex combination of existing movements. (We call this a ‘fit’ in Fig. 4; see Methods Section 1.10.) We then use the same combination of the associated library of gain patterns to construct a new gain pattern (see Fig. 4a). Interestingly, the resulting network output closely resembles the target movement (see Fig. 4b). This may seem unintuitive, but one can understand this result mathematically by calculating power-series expansions of the solution of the linearized neuronal dynamics (see Supplementary Math Note).

Finally, increasing the number of elements in the movement library reduces the error between a target movement and its fit, which is also reflected in a progressively better match between the target and the network output (see Figs. 4b–d and Supplementary Fig. 3). Although the idea of using motor primitives to facilitate rapid acquisition of new movements is well established [26, 25], our approach proposes the first (to our knowledge) circuit-level mechanism for achieving this objective. In addition to neuromodulatory systems [19, 20, 22], the cerebellum is a natural candidate structure to coordinate such motor primitives [25], as it is known to project to M1 and to play a critical role in error-based motor learning [35, 25].

Nonlinear behaviour

We initially choose the baseline firing rate (r_0 = 20 Hz in Eqn. (2)) to be consistent with experimentally measured firing rates in motor cortex [4, 36, 28]. Most of the time, neurons operate within the linear part of their nonlinear gain function (i.e., the neuronal dynamics are similar to the case of using the linear gain function $f(x_i; g_i) = g_i\, x_i$ (see Figs. 5a,c)). To test if our results hold for scenarios with more strongly nonlinear dynamics, we reduce the baseline firing rate to r_0 = 5 Hz. This increases the neuronal activity near the lower-saturation regime (i.e., towards the left part of the curve in the left panel of Fig. 1c) of the gain function (see Figs. 5b,c). As expected from the larger range of possible network outputs (and improved learning performance) in nonlinear recurrent neuronal networks than in linear ones [34, 31, 32], we observe better learning performance for r_0 = 5 Hz than for r_0 = 20 Hz (compare the black and blue curves in Fig. 5d), and we obtain a very similar distribution of gain values after training (see Fig. 5e).

Importantly, it is still possible to learn new movements by using combinations of existing gain patterns. As before, performance is limited by the accuracy with which one can construct target movements as linear combinations of existing primitives. (See the correlations between network output errors and fit errors in Supplementary Fig. 4b.) Moreover, errors in network output decrease on average with increasing numbers of gain patterns in the movement library (see the orange curve in Fig. 5f), and the difference between the network output and corresponding fit remains small for all tested numbers of library elements (see the blue curve in Fig. 5f). However, reducing r_0 to sufficiently small values (that are below 5 Hz) does eventually lead to a deterioration in the effectiveness of gain patterns providing motor primitives for new movements.

Gain modulation can control movement speed

Thus far, we have demonstrated that simple (even coarse, group-based) gain modulation enables control of network outputs of the same, fixed duration. To control movements of different durations, motor networks must be able to slow down or speed up muscle outputs (i.e., change the duration of movements without affecting their shape). In line with recent experimental results [37, 38], we investigate if changing neuronal gains allows control of the speed of an intended movement (see Fig. 6a and Methods Section 1.10). We begin with a network of 400 neurons (with 40 random modulatory groups) that generates muscle activity that lasts approximately 0.5 s. We find that our learning rule can successfully train a network to generate a slower variant that lasts 5 times longer (see Fig. 6b and Supplementary Fig. 5a) than the original movement (see Methods Section 1.10).

In contrast to simply changing the single-neuron time constant τ — which uniformly scales the duration, but does not affect the shape of each neuron's activity — modifying neuronal gains to generate 'fast' and 'slow' output variants leads to changes in both the shape and duration of neuronal firing rates, in line with recent experimental findings [37]. Changing neuronal gains thus enables interactions between the shape and duration of outputs without requiring retraining of the synaptic weight matrix to scale the duration of neuronal activities [39].

The learned slow variants are more sensitive to noisy initial conditions than the fast variants, but we can find more robust solutions by using a regularized back-propagation algorithm to train both the neuronal gains and the readout weights (see Methods Section 1.10). Following training, the slow variants are learned successfully (see Fig. 6c) and are less sensitive to the same noisy initial conditions (see Supplementary Fig. 5g). The neuronal dynamics oscillate transiently, with a substantially lower frequency than either the fast variants or the slow variants trained by our reward-based learning rule. (Compare the bottom panels of Fig. 6c and Fig. 6b.) We also find a single gain pattern that, rather than slowing down only one movement, slows down up to approximately five distinct movements, which result from five orthogonal initial conditions, by a factor of 5 (see Supplementary Figs. 5h–j). Consequently, one can extend the temporal scale of transient neuronal activity several-fold through specific changes in neuronal gains.

Smoothly controlling the speed of movements

Following training on a fast and slow variant of the same movement (see the previous section), we find that naively interpolating between the two gain patterns does not yield the same movement at intermediate speeds (see the top panel of Fig. 6d), consistent with human subjects being unable to consistently apply learned movements at novel speeds [39, 40]. Therefore, even when we consider 'fast' and 'slow' variants of the same movement, both our learning rule and the back-propagation training do not learn to 'slow down' the movement; instead, they learn two seemingly unrelated gain patterns. However, it is possible to modify our back-propagation training procedure by including additional constraints on the fast and slow gain patterns (see Methods Section 1.10) so that interpolating between the two gain patterns produces progressively faster or slower outputs. We successfully train the network to generate two movements (associated with two different initial conditions) at 7 different speeds with durations that range from 0.5 s to 2.5 s (see Fig. 6e, Supplementary Fig. 6, and Methods Section 1.10). Linear interpolation between the fast and slow gain patterns (see Supplementary Fig. 6b) now generates smooth speed control of both movements at any intermediate speed (see the bottom panel of Fig. 6d as well as Fig. 6f). In other words, we can control the speed of multiple movements associated with different initial conditions by learning a ‘manifold’ [41] in neuronal gain space that interpolates between the fast and the slow gain patterns (see the bottom right of Fig. 6a).

Joint control of movement shape and speed

Thus far, we have shown that gain modulation can affect either the shape or the speed of a movement. Flexible and independent control of both the shape and speed of a movement (i.e., joint control) necessitates separate representations of space and time in the gain patterns. A relatively simple possibility is to find a single universal manifold in neuronal gain space (see the previous section) for speed control (we call this the ‘speed manifold’) and combine it with gain patterns that are associated with different movement shapes. Biologically, this may be achievable using separate modulatory systems. We achieve such separation by simultaneously training one speed manifold and 10 gain patterns for 10 different movement shapes such that movements are encoded by the product of shape-specific and speed-specific gain patterns. (See Fig. 7a and Methods Section 1.10.) Following training, we can generate each of the 10 movements at the 7 trained speeds by multiplying a speed-specific gain pattern (see Fig. 7b) with the desired shape-specific gain pattern. Importantly, we can also accurately generate each of the 10 different movements at any intermediate speed by simply linearly interpolating between the fast and slow gain patterns (see Figs. 7c,d). We thereby obtain separate families of gain patterns for movement shape and speed that independently control movements in space and time.

Learning gain-pattern primitives to control movement shape and speed

To construct new movement shapes with arbitrary durations, we examine the possibility of using both the speed manifold and the 10 trained shape-specific gain patterns that we obtained previously (see Fig. 7) as a library of spatiotemporal motor primitives. We test this library using 100 novel target movement shapes (see Fig. 4). For each target movement, we learn the coefficients for linearly combining the 10 shape-specific gain pattern primitives to obtain each new movement at both the fast and slow speeds while keeping the speed manifold fixed (see Fig. 8a and Methods Section 1.10).

We find that it is possible to accurately generate the new movements at fast and slow speeds using the above spatiotemporal library of gain patterns (see Supplementary Fig. 7), and we are able to produce the new movements with similar accuracies as those at the fast and slow speeds at any intermediate speed by linearly interpolating between the fast and slow gain patterns from the unaltered speed manifold. (See Fig. 8b and the black and red curves in Fig 8c.) The mean error of approximately 0.5 across all movement durations is similar to the error that we obtained previously from a movement library that consists of 10 gain patterns (see Fig. 4d). We can substantially outperform both the (uniformly-at-random) permuted gain patterns from their associated targets (see Methods Section 1.10) and using least-squares fitting (which we used previously) to combine gain patterns. (See the grey and black dashed curves in Fig. 8c.)

Consistent with the idea of rapidly generating movements using motor primitives, we generate correlated target shapes by using correlated combinations of gain patterns (see Fig.

8d). Therefore, one can use previously learned gain patterns for controlling movement shapes to generate new movements while maintaining independent control of movement speed.

Discussion

The movement-specific population activity that has been observed in monkey primary motor cortex [4], can arise through several possible mechanisms. Distinct neuronal activity can emerge from a fixed population-level dynamical system with different movement-specific preparatory states [7]. Alternatively, one can change the underlying dynamical system through modification of the effective connectivity [42] even when a preparatory state is the same across movements. Such changes in effective connectivity can arise either through a feedback loop (e.g., a low-rank addition to the synaptic weight matrix [34]) or through patterns of movement-specific gains, as we explored in this paper. We found that movement-specific gain patterns provide a similar performance to training a different initial condition for each desired output (with a fixed duration) and that both of these approaches outperform a rank-1 perturbation of the synaptic weight matrix (see Supplementary Fig. 1f). Gain modulation thus provides a complementary method of controlling neuronal dynamics for flexible and independent manipulation of output shape. Additionally, gain modulation provides a compelling mechanism for extending the duration of activity transients without needing to carefully construct movement-specific network architectures [39].

Gain modulation may occur via neuromodulators [20, 22], but it can also arise from a tonic (i.e., static) input that shifts each neuron's resting activity within the dynamic range of its input–output function (for example, through inputs from the cerebellum) [14]. Although this is an effective way of mimicking gain changes in recurrent network models with strongly nonlinear single-neuron dynamics [37, 43], we were unable to produce desired target outputs by training a tonic input. It is worth noting that a tonic input also modifies baseline neuronal activity, thereby altering the output muscle activities away from rest.

In line with previous research [8, 4, 10], we trained networks to generate specific target output trajectories (which we suggest act as a proxy for muscle activity). This is a simplification of actual motor learning, as there are many different possible muscle activations that can lead to a 'successful' movement. For some motor tasks, it is probably more biologically plausible to train a network to increase the success of the desired movement defined by the position of an end effector while also minimizing the total amount of muscle activity (e.g., see [32, 44]). Nevertheless, our learning rule is biologically plausible, in that it uses only local information and a single scalar signal (which is the total sum of squared errors) per trial. It does not carry detailed information about the exact way in which an output trajectory deviates from a desired trajectory. We thus expect that our main results will still be relevant for more realistic models of motor learning (e.g., using a biophysically realistic model of a human arm [32]).

In our model, in which the recurrent architecture remains fixed, synaptic modifications may take place upstream of the motor circuit (e.g., in the input synapses to the presumed neuromodulatory neurons [45]). Additionally, changes in neuronal gains can work in concert with synaptic plasticity in cortical circuits, thereby allowing changes in the modulatory state of a network to be transferred into circuit connectivity [46], consistent with known interactions between neuromodulation and plasticity [45]. Consequently, understanding the neural basis of motor learning may necessitate recording from a potentially broader set of brain areas than those circuits whose activity correlates directly with movement dynamics.

Our results build on a growing literature of taking a dynamical-systems approach to studying temporally-structured cortical activity. This perspective has been effective for investigations of several cortical regions [5, 37, 7, 36, 47, 4, 48]. In line with this approach, our results may also be applicable to other recurrent cortical circuits that exhibit rich temporal dynamics (e.g., decision-making dynamics in prefrontal cortex [48], temporally-structured memories, etc.).

In summary, our results support the view that knowing only the structure of neuronal networks is not sufficient to explain their dynamics [49, 50]. We extend current understanding of the effects of neuromodulation [17, 20, 49, 13] and show that it is possible to control a recurrent neuronal network's computations without changing its connectivity. We found that modulating only neuronal responsiveness enables flexible control of neuronal activity. We were also able to combine previously learned modulation states to generate new desired activity patterns, and we demonstrated that employing gain modulation allows one to smoothly and accurately control the duration of network outputs. Our results thus suggest the possibility that gain modulation is a central part of motor control.

1 Methods

Our model is specified by a differential equation governing the neuronal firing rates (Eqn. (1)), the gain function Eqn. (2), a set of readout weights, and each neuron's gain. In the following, we describe our model precisely.

1.1 Neuronal dynamics

We model neuronal activity according to Eqn. (1), which we integrate using the ODE45 function (using default parameters) in MATLAB. We do not explicitly model dynamics prior to movement execution; all of our simulations begin at the time of movement onset [11, 4] (except when we use a ramping input in Fig. 2). We choose the initial condition $\boldsymbol{x}_0$ among the 'most observable' modes of the system (i.e., those that elicit the strongest transient dynamics [11]). Specifically, we first linearize the dynamics Eqn. (1) around its unique equilibrium point $\boldsymbol{x} = \boldsymbol{0}$ using unit gains (i.e., $g_i = 1$ for all i), and we compute the observability Gramian (a symmetric positive-definite matrix $\boldsymbol{Q}$) of the linearized system. The most observable modes are the top eigenvectors of $\boldsymbol{Q}$ [11]. Unless we state otherwise, we choose the eigenvector associated with the largest eigenvalue of $\boldsymbol{Q}$ (note that all of its eigenvalues are real and positive) as the initial condition $\boldsymbol{x}_0$ for the neuronal activity. Following [11], we also scale $\boldsymbol{x}_0$ so that $||\boldsymbol{x}_0||_2 = 1.5\sqrt{N}$.

1.2 Biophysical interpretation of Eqn. (1)

Equation (1), together with Eqn. (2), describes how we model neuronal firing rates relative to a baseline rate r_0. In this section, we clarify that one can obtain identical neuronal activity by using a strictly positive gain function f and including a constant input $\boldsymbol{h}$ in Eqn. (1). Specifically, given a desired baseline firing rate r_0, one can model the neuronal activity as

$$\tau \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \boldsymbol{W} f(\boldsymbol{x}(t); \boldsymbol{g}) + \boldsymbol{h} \quad (3)$$

for the same initial condition $\boldsymbol{x}_0$ that we described above, where $h_i = -r_0 \Sigma_j W_{ij}$ and

$$f(x_i; g_j) = \begin{cases} r_0 \tanh(g_i x_i / r_0) + r_0, & \text{if } x_i < 0, \\ (r_{\max} - r_0) \tanh(g_i x_i / (r_{\max} - r_0)) + r_0, & \text{if } x_i \geq 0, \end{cases} \quad (4)$$

where $r_{\max}$ is the maximum firing rate. Note that the constant term $\boldsymbol{h}$ in Eqn. (3) is necessary to balance the additional r_0 term in Eqn. (4).

1.3 Construction of the network architecture

Prior to optimization, we generate synaptic weight matrices $\boldsymbol{W}$ as detailed in Ref. [11]. In keeping with Dale's law, these matrices consist of M positive (excitatory) columns and M negative (inhibitory) columns. We begin with a set of sparse (such that the connection probability between any two neurons is small) and strong weights with nonzero elements set to $w_0/\sqrt{N}$ (excitatory) and $-\gamma w_0/\sqrt{N}$ (inhibitory), where $w_0^2 = 2\rho^2/\left(p(1-p)\left(1+\gamma^2\right)\right)$ and the connection probability between each two neurons is homogeneous and is given by $p = 0.1$. This construction results in $\boldsymbol{W}$ having an approximately circular spectrum (i.e., set of eigenvalues) of radius ρ (which we set to $\rho = 10$), leading to linear instability before stability optimization (see below). As in Ref. [11], we set the inhibition/excitation ratio γ to be $\gamma = 3$.

After constructing the initial $\boldsymbol{W}$, we never change any of the excitatory connections. Following [11], we refine the inhibitory connections to minimize an upper bound of $\boldsymbol{W}$'s 'spectral abscissa' (SA) (i.e., the largest real part among the eigenvalues of $\boldsymbol{W}$) [11]. Briefly, we iteratively update inhibitory weights to follow the negative gradient of this upper bound to the SA. First, the inhibitory weights remain inhibitory (i.e., negative). Second, we maintain a constant ratio (of $\gamma = 3$) of mean inhibitory weights to mean excitatory weights. Third, we restrict the density of inhibitory connections to be less than or equal to 0.4 to maintain sufficiently sparse connectivity. We observed that this constrained gradient descent usually converges within a few hundred iterations. As was noted in Ref. [11], the SA typically decreases during optimization from 10 to about 0.15. For additional details, see the supplemental information of Ref. [11].

As a proof of principle, we also construct a 'chaotic' variant of our recurrent neuronal-network model (see Fig. 3). These networks are chaotic in the sense that the neuronal dynamics in Eqn. (1) have a positive maximum Lyapunov exponent [51]. We use a synaptic weight matrix $\boldsymbol{W}$ (as described above) prior to optimization, but now use parameter values of $\gamma = 1$ and $\rho = 1.5$. We also set $\tau = 20$ ms, and we choose the initial condition $\boldsymbol{x}_0$ for the neuronal activity from a uniform distribution on the interval [−10,10]. We use only the first 0.5 s of neuronal activity for our simulations of the chaotic network model.

1.4 Creating target muscle activity

We generate target muscle activities of duration $t_{\mathrm{tot}} = 500$ ms (see Figs. 1–5) and $t_{\mathrm{tot}} = 2{,}500$ ms (see Figs. 6–8). In each case, we draw muscle activity from a Gaussian process with a covariance function $K \in [0, t_{\mathrm{tot}}] \times [0, t_{\mathrm{tot}}] \to \mathbb{R}_{\geq 0}$ that consists of a product of a squared-exponential kernel (to enforce temporal smoothness) and a non-stationary kernel that produces a temporal envelope similar to that of real electromyogram (EMG) data during reaching [4]. Specifically,

$$K(t, t') = e^{\frac{-(t-t')^2}{2\ell^2}} \times E(t/\sigma) \times E(t'/\sigma), \quad (5)$$

where $E(t) = te^{(-t^2/4)}$. We set $\sigma = 110$ ms and $\ell = 50$ ms for movements that last 500 ms, and $\sigma = 550$ ms and $\ell = 250$ ms for movements that last 2, 500 ms. We also multiply the resulting muscle activity by a scalar to ensure that it has the same order of magnitude as the neuronal activity. We use a sampling rate of 400 Hz for movements that last 500 ms and 200 Hz for movements that last 2, 500 ms.

We are modelling network output as a proxy for muscle-force activity. When we study whether we can generate the same movement that lasts 5 times longer (see Figs. 6–8), we scale the duration of the muscle activity without changing its amplitude. To actually generate the same movement so that it lasts 5 times longer, we also need to scale the amplitude of the muscle activity by the factor $1/5^2 = 1/25$. To demonstrate the effectiveness of learning through gain modulation, we omit this scaling, so the tasks on which we train are more difficult ones, as the target activity without the scaling has a substantially larger amplitude throughout the movement. However, we find that learning through gain modulation can also account for this scaling of muscle activity when performing movements at different speeds (see Supplementary Fig. 8). Alternatively, it may be possible for gain modulation of downstream motoneurons in the spinal cord to account for scaling of the amplitude of muscle activity when performing movements at different speeds (for example, see Ref. [21]).

1.5 Network output

We compute the network output $z(t)$ as a weighted linear combination of excitatory neuronal firing rates:

$$z(t) = \boldsymbol{m}^{\mathrm{T}} f\left(\boldsymbol{x}^{E}(t); \boldsymbol{g}^{E}\right) + b, \quad (6)$$

where $\boldsymbol{m}$, $\boldsymbol{x}^E(t)$, $\boldsymbol{g}^E \in \mathbb{R}^M$, the quantity $\boldsymbol{x}^E(t)$ is the excitatory neuronal activity, and M is the number of excitatory neurons. To ensure that the network output corresponds to realistic muscle activity (see Methods Section 1.4) prior to any training of the neuronal gains, we fit the readout weights $\boldsymbol{m}$ and the offset b to an initial output activity (see Methods Section 1.4) using least-squares regression. To ameliorate any issues of overfitting, we use 100 noisy trials, in which we add white Gaussian noise to the initial condition $\boldsymbol{x}_0$ for each trial with a signal-to-noise ratio of 30 dB [11]. Subsequently, the readout weights remain fixed throughout training of the neuronal gains. See our simulation details for each figure for additional details.

1.6 Measuring error in network output

We compute the error ε between the network output $\boldsymbol{z} \in \mathbb{R}^{t_{\text{tot}}}$ and a target $\boldsymbol{y} \in \mathbb{R}^{t_{\text{tot}}}$ by discretizing time and calculating

$$\varepsilon = 1 - R^2 = \frac{\sum_{t=1}^{t_{\text{tot}}} (z(t) - y(t))^2}{\sum_{t=1}^{t_{\text{tot}}} (y(t) - \bar{y})^2}, \quad (7)$$

where $\bar{y} = \frac{1}{t_{\text{tot}}} \sum_{t=1}^{t_{\text{tot}}} y(t)$ and R^2 is the coefficient of determination (which is often called simply ‘R-squared’). Therefore, an error of $\varepsilon = 1$ implies that the performance is as bad as if the output $\boldsymbol{z}$ were equal to the mean of the target $\boldsymbol{y}$ and thus does not capture any variations in output. When we use multiple readout units, we take the mean error ε across all outputs. We use this definition of error throughout the entire paper.

1.7 A learning rule for neuronal input–output gains

We devise a reward-based node-perturbation learning rule that is biologically plausible in the sense that it includes only local information and a single scalar reward signal that reflects a system’s recent performance [29, 30]. Our learning rule progressively reduces the error (on average) between the network output and a target output over training iterations. We update the gain g_i for neuron i after each training iteration t_n (with $n = 1,2,3, \ldots$) according to the following learning rule:

$$g_i(t_n) = g_i(t_{n-1}) + R(t_{n-1})(g_i(t_{n-1}) - \bar{g}_i(t_{n-1})) + \xi_i(t_n), \quad (8)$$

where

$$\begin{aligned} R(t_n) &= sgn(\bar{\varepsilon}(t_{n-1}) - \varepsilon(t_n)), \\ \bar{\varepsilon}(t_n) &= \alpha\bar{\varepsilon}(t_{n-1}) + (1-\alpha)\varepsilon(t_n), \\ \bar{g}_i(t_n) &= \alpha\bar{g}_i(t_{n-1}) + (1-\alpha)g_i(t_n), \end{aligned} \quad (9)$$

where $\varepsilon(t_n)$ represents the output error at iteration t_n (see Methods Section 1.6), *sgn* is the sign function, $\xi_i(t_n) \sim \mathcal{N}(0, 0.001^2)$ is a Gaussian random variable with mean 0 and standard deviation 0.001, and $\alpha = 0.3$. The initial modulatory signal is $R(t_0) = 0$, and the other initial conditions are $\bar{\varepsilon}(t_0) = \varepsilon(t_0)$ (where $\varepsilon(t_0)$ is the initial error before training) and $\bar{g}_i(t_0) = g_i(t_0) = 1$. One can interpret the terms $\bar{g}_i$ and $\bar{\varepsilon}$ as low-pass-filtered gains and errors, respectively, over recent iterations, with a history controlled by the decay rate α [32]. We use these parameter values in all of our simulations in this paper. We find that varying the standard deviation of the noise term ξ or the factor α has little effect on the learning dynamics (not shown), in line with Ref. [31].

Although our learning rule in Eqn. (8) is similar to reward-modulated 'exploratory Hebbian' (EH) synaptic plasticity rules [30, 31, 32], we investigate changes in neuronal gains (i.e., the responsiveness of neurons) inside a recurrent neuronal network, rather than synaptic weight changes. The above notwithstanding, we expect our learning rule to perform well for a variety of learning problems. For example, it can solve credit-assignment problems, because one can formulate such a node-perturbation learning rule as reinforcement learning with a scalar reward [52].

The modulatory signal R does not provide information about the sign and magnitude of the error, and it also does not indicate the amount that each readout (if using multiple readouts) contributes to a recent change in performance. The modulatory signal R indicates only whether performance is better or worse, on average, compared with previous trials. One can view the modulatory signal as an abstract model for phasic output of dopaminergic systems in the brain [53, 19, 24, 23].

We use the following procedure for updating neuronal gains. We update the gains for iteration t_1 according to Eqn. (8), and we obtain the network output from the gain pattern $\boldsymbol{g}(t_1)$. We then calculate the error $\varepsilon(t_1)$ from the output, and we subsequently calculate the modulatory signal $R(t_1)$ and the quantities $\bar{\varepsilon}(t_1)$ and $\bar{\boldsymbol{g}}(t_1)$ using Eqn. (9). We then repeat this process for all subsequent iterations. If any gain values become negative, we set these to 0. However, this happened very rarely in our computations, and we observed it only when we used 60,000 training iterations (i.e., in Figs. 3e and 6b).

1.8 Alternative learning rule

One can also adapt our learning rule so that learning ceases when the modulatory signal $R(t_n)$ saturates at a sufficiently small value. A way to achieve this is by instead placing the noise term ξ_i inside the brackets in Eqn. (8), so that the modulatory signal R multiplies ξ_i, together with changing the *sgn* function in Eqn. (9) to the *tanh* function. This yields the following learning rule:

$$g_i(t_n) = g_i(t_{n-1}) + R(t_{n-1})(g_i(t_{n-1}) - \bar{g}_i(t_{n-1}) + \xi_i(t_n)), \quad (10)$$

where

$$\begin{aligned} R(t_n) &= tanh(\eta(\bar{\varepsilon}(t_{n-1}) - \varepsilon(t_n))), \\ \bar{\varepsilon}(t_n) &= \alpha\bar{\varepsilon}(t_{n-1}) + (1-\alpha)\varepsilon(t_n), \\ \bar{g}_i(t_n) &= \alpha\bar{g}_i(t_{n-1}) + (1-\alpha)g_i(t_n), \end{aligned} \qquad (11)$$

and $\eta = 50{,}000$ controls the slope of the *tanh* function at 0 (i.e., when the low-pass-filtered error $\bar{\varepsilon}(t_n)$ matches the current error $\varepsilon(t_n)$). Learning now stops when $\bar{\varepsilon}(t_{n-1}) = \varepsilon(t_n)$; see the purple curve in Fig. 2a. We achieve a qualitatively similar learning performance by using Eqns. (10) and (11) instead of Eqns. (8) and (9), respectively. Compare the purple and red curves in Fig. 2a.

1.9 Generating groups for group-based gain modulation

For coarse-grained (i.e., grouped) gain modulation, we generate n (modulatory) groups, and we independently modulate each group using one external 'modulatory unit'. Our generation mechanism for random groups is as follows. For each of the n groups, we choose N/n neurons (where N is the total number of neurons in the network) uniformly at random without replacement. If n does not divide N, we assign the remaining neurons to groups uniformly at random.

When using specialized groupings (see Figs. 3b,c and Supplementary Figs. 2a–d) for a particular target movement, we obtain groups by applying k-means clustering (where k is the desired number of groups) to 10 gain patterns that we obtain from 10 prior independent training sessions (using neuron-specific control) on the same target and which correspond to the minimum error for each training session. We thus apply k-means clustering to a matrix of size $N \times 10$, where row i has the gain values for neuron i from the 10 independent training sessions to the same target. Applying k-means clustering then generates groupings in which neurons in the same group tend to have similar gain values following training using neuron-specific modulation.

1.10 Simulation details

We now give a brief summary about our simulations for Figures 1–8. See the Supplementary Math Note for our mathematical derivations and see our Full Simulation Details for further information. We also provide sample MATLAB code at http://modeldb.yale.edu/246004. Also see our Life Sciences Reporting Summary for additional information.

Refer to Web version on PubMed Central for supplementary material.

We train neuronal gains on the same task as the one that we showed in Fig. 1d using 3 alternative models. For one model, we use a ramping input to the neuronal activity in Eqn. (1) as a model of preparatory activity prior to movement onset [11, 4]. We use the same ramping input function as the one that was used in Ref. [11]. It is $\exp(t/\tau_{\text{on}})$ for $t < 0$ s and $\exp(-t/\tau_{\text{off}})$ after movement onset ($t \geq 0$), with an onset time of $\tau_{\text{on}} = 400$ ms and an offset time of $\tau_{\text{off}} = 2$ ms. Gain changes that result from learning now also affect the neuronal activity at $t = 0$ (i.e., at movement onset).

We also train a 'chaotic' [34] variant of our model (see Methods Section 1.3, where we describe how we construct such a model), and we use the first 0.5 s of neuronal activity.

Finally, we use an alternative learning rule (see Eqns. (10) and (11)) in which learning stops automatically when the difference between network output errors in successive training iterations becomes sufficiently small (see Methods Section 1.7).

For Figs. 3b,c, we generate 5 different target outputs and run 10 independent training sessions for each target. For the random groupings (see Methods Section 1.9), we use different independently-generated random groups for each simulation. For the specialized groups (see Methods Section 1.9), for a given number of groups, we use the same grouping in all simulations.

We now explain how we determine specialized groups that are shared by multiple movements (i.e., we use the same grouping for learning multiple movements); see the plots in Fig. 3c and Supplementary Figs. 2b–d. We apply k-means clustering (where k is the desired number of groups) across all of the gain patterns that we obtain using neuron-specific modulation for each of the movements. That is, we apply k-means clustering to a matrix of size $N \times 10 \cdot q$, where N is the number of neurons and q is the number of movements (and, equivalently, the number of gain patterns).

For the task that we just described above, we consider various different numbers of groups (using random groupings) for networks with $N = 100$, $N = 200$, and $N = 400$ neurons. We again perform 10 independent training sessions for each network, target, and number of groups. We fit the readout weights so that each scenario generates the same network output when all gains are set to 1. The readout weights remain fixed throughout training. We plot these results in Fig. 3d and Supplementary Figs. 2e–h.

When we use multiple readout units, we generate 10 different initial and target outputs for each readout unit. We run independent training sessions for these 10 sets of target outputs and calculate mean errors across the 10 training sessions. For a given number of readout units, we use the same sets of initial and target outputs for all 3 network sizes and each number of random modulatory groups. We thus fit readout weights so that each scenario generates the same output with all gains set to 1. The readout weights remain fixed throughout training. We use 60,000 (instead of 18,000) training iterations to ensure error saturation.

To create libraries of learned movements, we train a network of 400 neurons and 40 random groups (see Methods Section 1.9) on each of 100 different target movements independently. (In other words, this generates 100 different gain patterns, with one for each movement.) For library sizes of $l \in \{1,2,\ldots,20\}$, we choose 100 samples of l movements (from the learned gain patterns and their outputs) uniformly at random without replacement for each l. We then fit the set of l movements in each of the 100 sample libraries using least-squares regression for each of 100 hitherto-untrained novel target movements. We constrain the fitting coefficients c_j from the least-squares regression by requiring that $c_j \geq 0$ for all j and $\sum_{j=1}^{l} c_j = 1$. We calculate the fit error (i.e., the error between the fit and the target), the output error (i.e., the error between the output and the target), and the error between the fit and the output for each of the 100 novel target movements, each of the 100 library samples, and each l.

We train the same 200-neuron weight matrix that we used in Fig. 1 on the same task as the one that we showed in Figs. 1d–f, except with a baseline rate of $r_0 = 5$ Hz in Eqn. (2). We also repeat the simulations that we performed in Fig. 4 for the baseline rate $r_0 = 5$ Hz.

In each of these simulations, we use a network of 400 neurons and 40 random modulatory groups (see Methods Section 1.9). We construct ‘slow’ (2.5 s) target movements with $\sigma =$ 550 ms and $\ell = 250$ ms in Eqn. (5). We then construct a ‘fast’ (0.5 s) variant of each movement. Each movement variant has 500 evenly-spaced points (see Methods Section 1.4). We sample the fast variant using 100 evenly-spaced points, and we then augment 400 instances of 0 values to the final 2 s of the movement to ensure that both movement variants have the same length.

For Fig. 6b, we fit readout weights using least-squares regression, such that with all gains set to 1, the network output generates the fast variant. We then train gain patterns using our learning rule in Eqns. (8) and (9) so that the network output generates the slow-movement variant. (The initial condition $\boldsymbol{x}_0$ and readout weights remain fixed.) We use 60,000 training iterations, and we run 10 independent training sessions for each of 10 different target movements.

For Fig. 6c, we perform the task that we described in the paragraph above using a gradient-descent training procedure with gradients that we obtain from back-propagation [54]. Together with learning the gain pattern for the slow variant, we jointly optimize a single set of readout weights (shared by both the fast-movement and slow-movement variants) (see Methods Section 1.5) as part of the same training procedure. The gains are still fixed at 1 for the fast variant. The cost function for the training procedure is equal to the squared Euclidean 2-norm between actual network outputs and the corresponding target outputs both at fast and slow speeds plus the Euclidean 2-norm of the readout weights, where the latter acts as a regularizer. We run gradient descent for 500 iterations, which is well after the cost has stopped decreasing.

For each of the 10 trained movements that we described earlier in this section, we extract the mean minimum error across all simulations for both the outputs obtained via our learning rule (see Supplementary Fig. 5a) and the outputs obtained via back-propagation (see Supplementary Fig. 5b). We then linearly interpolate between the learned gain patterns for the fast and slow outputs, and we calculate the error between the output and the target movement at the interpolated speed. (See the top panel of Fig. 6d.)

For Figs. 6d–f, we train networks to generate a pair of target movements in response to a corresponding pair of orthogonal initial conditions at fast and slow speeds and also at each of 5 intermediate, evenly-spaced speeds in between these extremes. To do this, we parametrize the gain pattern of speed index s (with $s \in \{1, \ldots, 7\}$) as a convex combination of a gain pattern $\boldsymbol{g}_{s=1}$ for fast movements and a gain pattern $\boldsymbol{g}_{s=7}$ for slow movements, with interpolation coefficients of λ_s (with $\boldsymbol{g}_s = \lambda_s \boldsymbol{g}_{s=1} + (1 - \lambda_s)\boldsymbol{g}_{s=7}$, $\lambda_1 = 1$, and $\lambda_7 = 0$). We optimize (using back-propagation, as discussed above) over $\boldsymbol{g}_{s=1}$, $\boldsymbol{g}_{s=7}$, the 5 interpolation coefficients λ_s (with $s \in \{2, \dots, 6\}$), and a single set of readout weights. For a given speed s, we use the gain pattern $\boldsymbol{g}_s$ for both movements. We call the collection of gain patterns $\boldsymbol{g}_s$ for $s \in \{1,\dots, 7\}$ the gain manifold for speed control (or the 'speed manifold', as a shorthand).

We train (using back-propagation) a 400-neuron network with 40 random modulatory groups (see Methods Section 1.9) to generate each of 10 different movement shapes at 7 different, evenly-spaced speeds (ranging from the fast variant to the slow variant) using a fixed initial condition $\boldsymbol{x}_0$. To jointly learn gain patterns that control movement shape and speed, we parametrize each gain pattern as the element-wise product of a gain pattern that encodes shape (which we use at each speed for a given shape) and a gain pattern that encodes speed (which we use at each shape for a given speed). We again parametrize (see our simulation details for Fig. 6) the gain pattern that encodes speed index s (with $s \in \{1,\dots, 7\}$) as a convex combination of two common endpoints, $\boldsymbol{g}_{s=1}$ (which we use for the fast-movement variants) and $\boldsymbol{g}_{s=7}$ (which we use for the slow-movement variants). We thus optimize over 10 gain patterns for movement shape, 2 gain patterns each for fast and slow movement speeds, 5 speed-interpolation coefficients, and a single set of readout weights.

In Fig. 7c, we calculate the mean error between the network output and the target over the 10 target movements when generating gain patterns for movement speed by linearly interpolating between the trained fast ($\boldsymbol{g}_{s=1}$) and slow ($\boldsymbol{g}_{s=7}$) gain patterns.

We use the 10 trained gain patterns for movement shapes, as well as the speed manifold from Fig. 7 (see our simulation details for Fig. 7). Using our learning rule from Eqns. (8) and (9), we train the 10 coefficients $c_1,\dots,c_{10}$ (see Fig. 8a) to construct a new gain pattern that, together with the speed manifold, generates a new target movement at the fast and slow speeds. Specifically, we replace the gains g_i (for $i \in \{1,\dots,N\}$) with the coefficients c_i (for $i \in \{1,\dots, 10\}$) in Eqns. (8) and (9). We use the mean of the errors at the fast and slow speeds in the learning rule. To generate the network output at the fast and slow speeds, respectively, we calculate the element-wise product between the newly-constructed gain pattern and the fast and slow gain pattern, respectively, on the speed manifold. We independently train, using 10,000 training iterations, the coefficients $c_1,\dots,c_{10}$ on each of the 100 target movements that we used for Fig. 4. As a control, we calculate the mean error between the network output and the target over the 100 target movements when choosing one of the 100 newly-learned gain patterns uniformly at random without replacement. (See the grey curve in Fig. 8c.)

Additionally, instead of learning to combine gain patterns using the method that we described in the previous paragraph, we determine coefficients $c_1,\dots,c_{10}$ using a least-squares regression by fitting the 10 learned movements to each of the 100 target movements at the fast and slow speeds simultaneously and requiring that $c_j \geq 0$ for all j and $\sum_{j=1}^{10} c_j = 1$. (See the black dashed curve in Fig. 8c.)

In Fig. 8d, we plot the Pearson correlation coefficient between pairs of target movements versus the Pearson correlation coefficient between corresponding pairs of learned coefficients $c_1,\ldots,c_{10}$. In our visualization, we plot only 1,000 of the 4,950 data points. (We choose these points uniformly at random.)

1.11 Statistics

The only statistical test that we use is a (nonparametric) paired Wilcoxon signed rank one-sided test in Supplementary Fig. 1e. No statistical methods were used to pre-determine sample sizes for our simulations, but our sample sizes are similar to those reported in previous studies (e.g., see [10, 11]). There was no randomization in our study because it was a computational study (we had no samples/organisms/participants in our study).

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgements

We thank the members of the Vogels lab (particularly E. J. Agnes, R. P. Costa, W. F. Podlaski, and F. Zenke) for their insightful comments and Y. T. Kimura for creating the monkey illustration. We also thank O. Barak, T. E. J. Behrens, R. Bogacz, M. Jazayeri, and L. Susman for their helpful comments. Our work was supported by grants from the Wellcome Trust (TPV and JPS through WT100000, and GH through 202111/Z/16/Z) and the Engineering and Physical Sciences Research Council through the Life Sciences Interface Doctoral Training Centre (University of Oxford) (EP/F500394/1) (JPS).

References

[1]. Rathelot J-A, Strick PL. Subdivisions of primary motor cortex based on cortico-motoneuronal cells. Proceedings of the National Academy of Sciences. 2009; 106(3):918–923.

[2]. Rosenbaum DA. Human Motor Control. Cambridge, USA: Academic Press; 2009.

[3]. Sanes JN, Donoghue JP. Plasticity and primary motor cortex. Annual Review of Neuroscience. 2000; 23(1):393–415.

[4]. Churchland MM, Cunningham JP, Kaufman MT, Foster JD, Nuyujukian P, Ryu SI, Shenoy KV. Neural population dynamics during reaching. Nature. 2012; 487(7405):1–8.

[5]. Shenoy KV, Sahani M, Churchland MM. Cortical control of arm movements: a dynamical systems perspective. Annual Review of Neuroscience. 2013; 36:337–359.

[6]. Afshar A, Santhanam G, Yu BM, Ryu SI, Sahani M, Shenoy KV. Single-trial neural correlates of arm movement preparation. Neuron. 2011; 71(3):555–564. [PubMed: 21835350]

[7]. Churchland MM, Cunningham JP, Kaufman MT, Ryu SI, Shenoy KV. Cortical preparatory activity: Representation of movement or first cog in a dynamical machine? Neuron. 2010; 68(3): 387–400. [PubMed: 21040842]

[8]. Russo AA, Bittner SR, Perkins SM, Seely JS, London BM, Lara AH, Miri A, Marshall NJ, Kohn A, Jessell TM, Abbott LF, et al. Motor cortex embeds muscle-like commands in an untangled population response. Neuron. 2018; 97(4):1–14. [PubMed: 29301096]

[9]. Churchland MM, Cunningham JP. A dynamical basis set for generating reaches. Cold Spring Harbor Symposia on Quantitative Biology. 2014; 79:67–80. [PubMed: 25851506]

[10]. Sussillo D, Churchland MM, Kaufman MT, Shenoy KV. A neural network that finds a naturalistic solution for the production of muscle activity. Nature Neuroscience. 2015; 18(7):1025–1033. [PubMed: 26075643]

[11]. Hennequin G, Vogels TP, Gerstner W. Optimal control of transient dynamics in balanced networks supports generation of complex movements. Neuron. 2014; 82(6):1394–1406. [PubMed: 24945778]

[12]. Sehgal M, Song C, Ehlers VL, Moyer JR. Learning to learn Intrinsic plasticity as a metaplasticity mechanism for memory formation. Neurobiology of Learning and Memory. 2013; 105:186–199. [PubMed: 23871744]

[13]. Kida H, Mitsushima D. Mechanisms of motor learning mediated by synaptic plasticity in rat primary motor cortex. Neuroscience Research. 2017:1–5.

[14]. Chance FS, Abbott LF, Reyes AD. Gain modulation from background synaptic input. Neuron. 2002; 35(4):773–782. [PubMed: 12194875]

[15]. Swinehart CD, Bouchard K, Partensky P, Abbott LF. Control of network activity through neuronal response modulation. Neurocomputing. 2004; 58:327–335.

[16]. Zhang J, Abbott LF. Gain modulation of recurrent networks. Neurocomputing. 2000; 32:623–628.

[17]. Marder E. Neuromodulation of neuronal circuits: Back to the future. Neuron. 2012; 76(1):1–11. [PubMed: 23040802]

[18]. Salinas E, Thier P. Gain modulation: A major computational principle of the central nervous system. Neuron. 2000; 27(1):15–21. [PubMed: 10939327]

[19]. Molina-Luna K, Pekanovic A, Rohrich S, Hertler B, Schubring-Giese M, Rioult-Pedotti MS, Luft AR. Dopamine in motor cortex is necessary for skill learning and synaptic plasticity. PloS One. 2009; 4(9):e7082. [PubMed: 19759902]

[20]. Thurley K, Senn W, Lüscher H-R. Dopamine increases the gain of the input–output response of rat prefrontal pyramidal neurons. Journal of Neurophysiology. 2008; 99(6):2985–2997. [PubMed: 18400958]

[21]. Vestergaard M, Berg RW. Divisive gain modulation of motoneurons by inhibition optimizes muscular control. Journal of Neuroscience. 2015; 35(8):3711–3723. [PubMed: 25716868]

[22]. Wei K, Glaser JI, Deng L, Thompson CK, Stevenson IH, Wang Q, Hornby TG, Heckman CJ, Körding KP. Serotonin affects movement gain control in the spinal cord. Journal of Neuroscience. 2014; 34(38):12690–12700. [PubMed: 25232107]

[23]. Hosp JA, Pekanovic A, Rioult-Pedotti MS, Luft AR. Dopaminergic projections from midbrain to primary motor cortex mediate motor skill learning. Journal of Neuroscience. 2011; 31(7):2481–2487. [PubMed: 21325515]

[24]. Huntley GW, Morrison JH, Prikhozhan A, Sealfon SC. Localization of multiple dopamine receptor subtype mRNAs in human and monkey motor cortex and striatum. Molecular Brain Research. 1992; 15(3–4):181–188. [PubMed: 1331674]

[25]. Thoroughman KA, Shadmehr R. Learning of action through adaptive combination of motor primitives. Nature. 2000; 407(6805):742–747. [PubMed: 11048720]

[26]. Giszter SF. Motor primitives — New data and future questions. Current Opinion in Neurobiology. 2015; 33:156–165. [PubMed: 25912883]

[27]. Rajan K, Abbott LF, Sompolinsky H. Stimulus-dependent suppression of chaos in recurrent neural networks. Physical Review E. 2010; 82(1):011903.

[28]. Lara AH, Cunningham JP, Churchland MM. Different population dynamics in the supplementary motor area and motor cortex during reaching. Nature Communications. 2018; 9:2754.

[29]. Mazzoni P, Andersen RA, Jordan MI. A more biologically plausible learning rule for neural networks. Proceedings of the National Academy of Sciences. 1991; 88(10):4433–4437.

[30]. Legenstein R, Chase SM, Schwartz AB, Maass W. A reward-modulated Hebbian learning rule can explain experimentally observed network reorganization in a brain control task. Journal of Neuroscience. 2010; 30(25):8400–8410. [PubMed: 20573887]

[31]. Hoerzer GM, Legenstein R, Maass W. Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. Cerebral Cortex. 2014; 24(3):677–690. [PubMed: 23146969]

[32]. Miconi T. Biologically plausible learning in recurrent neural networks for flexible decision tasks. eLife. 2017; 6:e20899. [PubMed: 28230528]

[33]. Li N, Chen T-W, Guo ZV, Gerfen CR, Svoboda K. A motor cortex circuit for motor planning and movement. Nature. 2015; 519(7541):51. [PubMed: 25731172]

[34]. Sussillo D, Abbott LF. Generating coherent patterns of activity from chaotic neural networks. Neuron. 2009; 63(4):544–557. [PubMed: 19709635]

[35]. Spampinato DA, Block HJ, Celnik PA. Cerebellar–M1 connectivity changes associated with motor learning are somatotopic specific. Journal of Neuroscience. 2017; 37(9):2377–2386. [PubMed: 28137969]

[36]. Kao JC, Nuyujukian P, Ryu SI, Churchland MM, Cunningham JP, Shenoy KV. Single-trial dynamics of motor cortex and their applications to brain-machine interfaces. Nature Communications. 2015; 6:7759.

[37]. Wang J, Narain D, Hosseini EA, Jazayeri M. Flexible timing by temporal scaling of cortical responses. Nature Neuroscience. 2018; 21(1):102–110. [PubMed: 29203897]

[38]. Soares S, Atallah BV, Paton JJ. Midbrain dopamine neurons control judgment of time. Science. 2016; 354(6317):1273–1277. [PubMed: 27940870]

[39]. Hardy NF, Goudar V, Romero-Sosa JL, Buonomano DV. A model of temporal scaling correctly predicts that Weber's law is speed-dependent. bioRxiv. 2017 159590.

[40]. Collier GL, Wright CE. Temporal rescaling of simple and complex ratios in rhythmic tapping. Journal of Experimental Psychology: Human Perception and Performance. 1995; 21(3):602–627. [PubMed: 7790836]

[41]. Gallego JA, Perich MG, Miller LE, Solla SA. Neural manifolds for the control of movement. Neuron. 2017; 94(5):978–984. [PubMed: 28595054]

[42]. Friston KJ. Functional and effective connectivity: A review. Brain Connectivity. 2011; 1(1):13–36. [PubMed: 22432952]

[43]. Sussillo D, Barak O. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. Neural Computation. 2013; 25(3):626–649. [PubMed: 23272922]

[44]. Kambara H, Shin D, Koike Y. A computational model for optimal muscle activity considering muscle viscoelasticity in wrist movements. Journal of Neurophysiology. 2013; 109(8):2145–2160. [PubMed: 23324321]

[45]. Martins ARO, Froemke RC. Coordinated forms of noradrenergic plasticity in the locus coeruleus and primary auditory cortex. Nature Neuroscience. 2015; 18(10):1483–1492. [PubMed: 26301326]

[46]. Swinehart CD, Abbott LF. Supervised learning through neuronal response modulation. Neural Computation. 2005; 17(3):609–631. [PubMed: 15802008]

[47]. Breakspear M. Dynamic models of large-scale brain activity. Nature Neuroscience. 2017; 20(3): 340–352. [PubMed: 28230845]

[48]. Mante V, Sussillo D, Shenoy KV, Newsome WT. Context-dependent computation by recurrent dynamics in prefrontal cortex. Nature. 2013; 503(7474):78–84. [PubMed: 24201281]

[49]. Bargmann CI. Beyond the connectome: How neuromodulators shape neural circuits. BioEssays. 2012; 34(6):458–465. [PubMed: 22396302]

[50]. Bassett DS, Sporns O. Network neuroscience. Nature Neuroscience. 2017; 20(3):353–364. [PubMed: 28230844]

[51]. Sompolinsky H, Crisanti A, Sommers HJ. Chaos in random neural networks. Physical Review Letters. 1988; 61(3):259–262. [PubMed: 10039285]

[52]. Saito H, Katahira K, Okanoya K, Okada M. Statistical mechanics of structural and temporal credit assignment effects on learning in neural networks. Physical Review E. 2011; 83(5)

[53]. Frémaux N, Gerstner W. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. Frontiers in Neural Circuits. 2016; 9:85. [PubMed: 26834568]

[54]. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986; 323(6088):533–536.

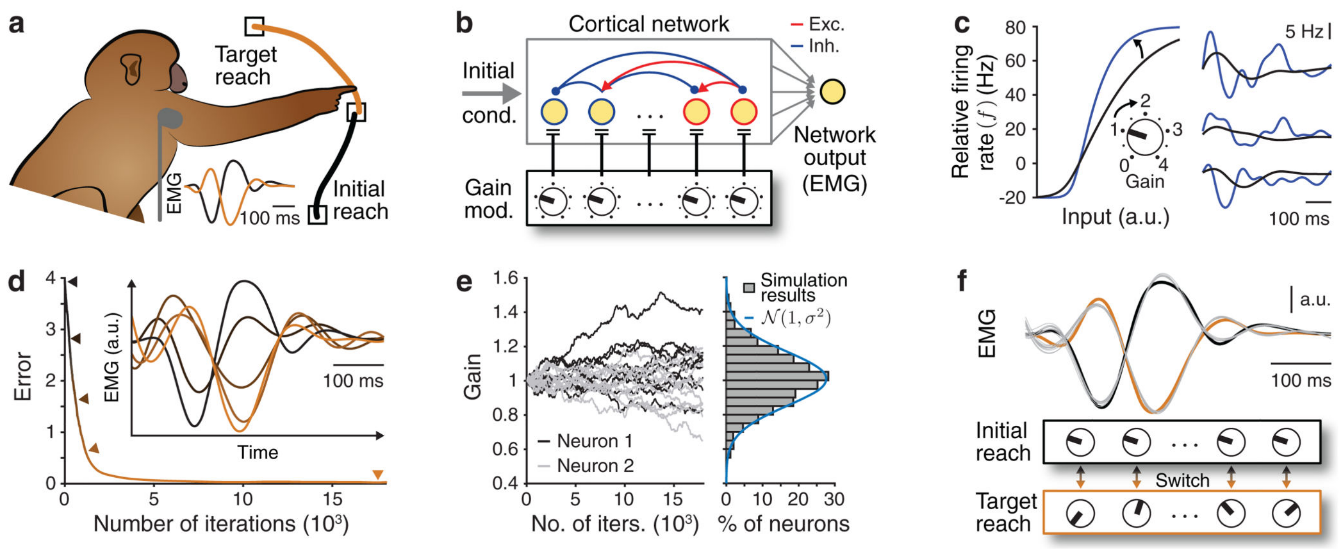

Figure 1. Controlling network activity through neuron-specific gain modulation.

(**a**) Example of a reaching task, with illustrative electromyograms (EMGs) of muscle activity for two reaches (in orange and black). (**b**) Schematic of our model (see the text and Methods Section 1.10). (**c**) (Left) Changing the slope of the input–output gain function uniformly for all neurons from (black) 1 to (blue) 2 has pronounced effects on (right) neuronal firing rates. We show results for three example neurons. (**d**) The mean error in network output decreases during training with neuron-specific modulation. In the inset, we show five snapshots of network output (indicated by arrowheads) as learning progresses. (**e**) (Left) Neuronal gain changes during training for 2 example neurons (grey and black) and 10 training sessions to the same target. (Right) Histogram of gain values after training. The blue curve is a Gaussian fit with a standard deviation of $\sigma \approx 0.157$. (**f**) Network outputs (grey curves) with all gains set to 1 and a new learned gain pattern for 10 noisy initial conditions compared to both targets (black and orange). (We use a 200-neuron network for all simulations in this figure.)

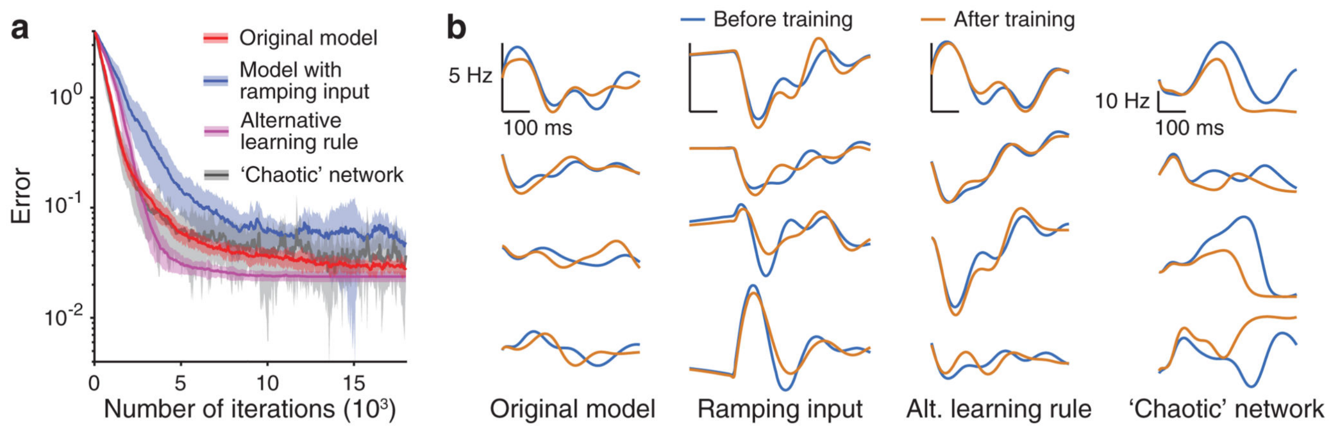

Figure 2. Learning through gain modulation in different models.

(**a**) Mean error over 10 independent training sessions for our original model that we used in Fig. 1d (red); the model with a biologically motivated ramping input (blue); the model when using the alternative learning rule Eqn. (10), in which learning automatically stops at a sufficiently small error (purple); and when using a 'chaotic' recurrent network model (grey) (see Methods Section 1.10). Shading indicates one standard deviation. (**b**) The firing rates of 4 example neurons before (i.e., with all gains set to 1) and after training the neuronal gains in (left) our original model, (centre left) our model with a ramping input, (centre right) our model with the alternative learning rule, and (right) the model when using a 'chaotic' network. (We use 200-neuron networks for all simulations in this figure.)

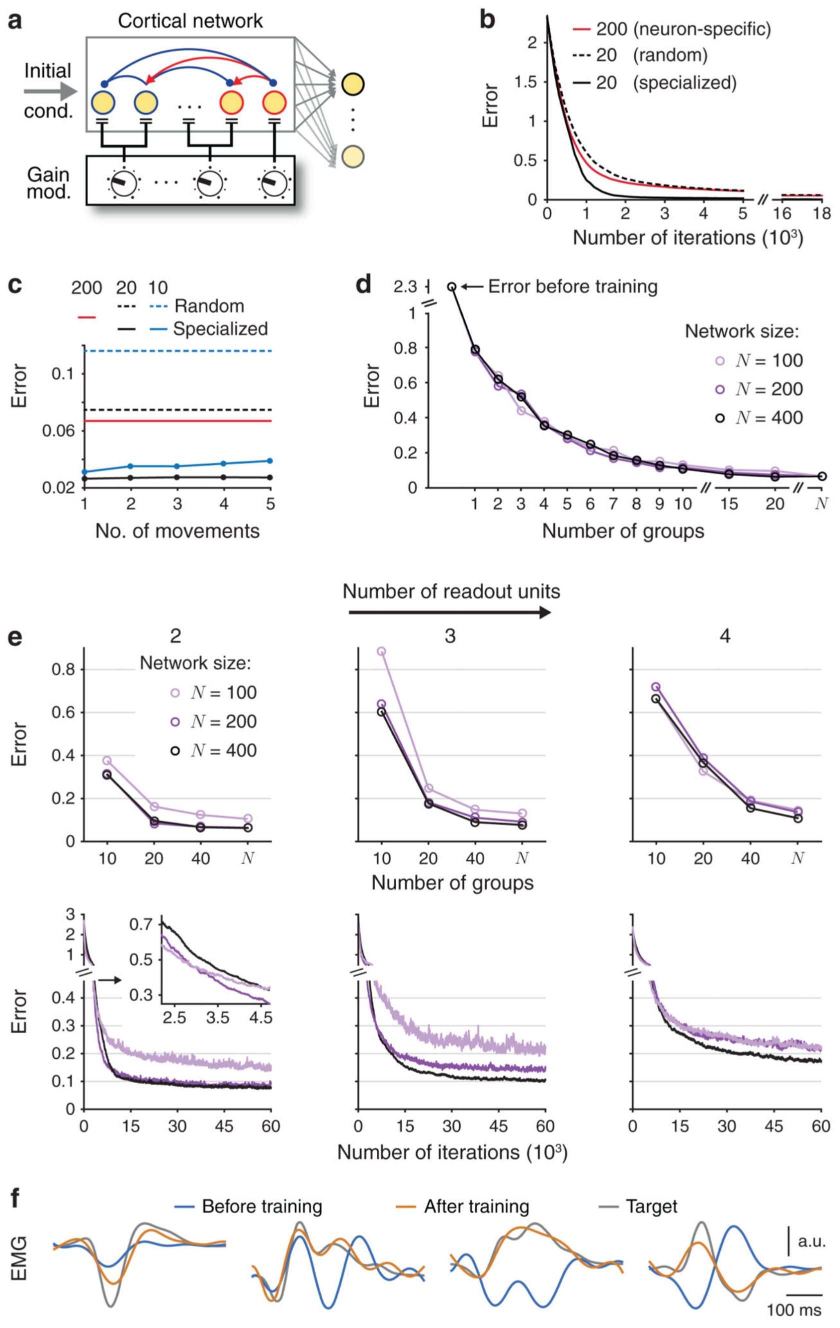

Figure 3. Controlling network activity through coarse, group-based gain modulation.

(**a**) We identically modulate neurons within each group (see Methods Section 1.9). Target outputs can involve multiple readout units. (**b**) Mean error during training for 20 random, 20 specialized, and 200 (i.e., neuron-specific) groups. (See Methods Section 1.10 for more details.) (**c**) Mean minimum errors after training using specialized groups. We use the same grouping for learning multiple different movements. (**d**) Mean minimum errors for different numbers of random groups with networks of 100, 200, and 400 neurons. (The N on the horizontal axis indicates neuron-specific modulation.) In panels (b)–(d), we use a single readout unit. (**e**) (Top) Mean minimum error as a function of the number of random groups when learning each of (left) 2, (centre) 3, and (right) 4 readouts for the same networks as in panel (d). (Bottom) The corresponding mean errors during training for the case of 40 groups. The inset is a magnification of the initial training period for the case of 2 readout units. (**f**) Outputs producing the median error for the case of 4 readout units using 40 groups in the 400-neuron network.

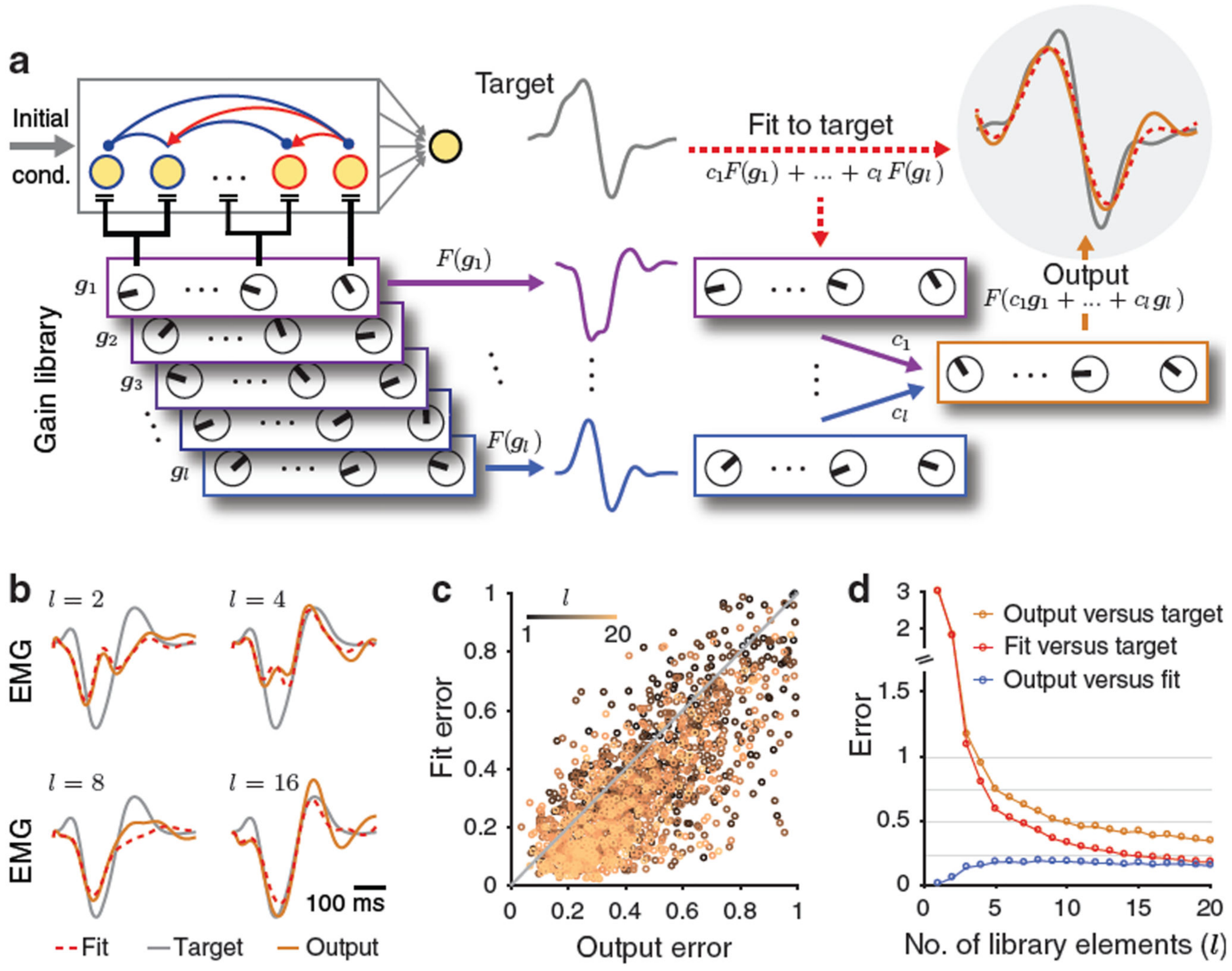

Figure 4. Gain patterns can provide motor primitives for novel movements.

(**a**) Schematic of a learned library of gain patterns ($\mathbf{g}_1,\ldots,\mathbf{g}_l$), which we colour from purple to blue) and a combination $c_1F(\mathbf{g}_1) +\ldots+ c_lF(\mathbf{g}_l)$ of their outputs (which we denote by F) that we fit (red dashed curve) to a novel target (grey curve). (Upper right) The output $F(c_1\mathbf{g}_1 +\ldots + c_l\mathbf{g}_l)$ (which we show in orange) of the same combination of corresponding gain patterns also closely resembles the target. We use a 400-neuron network with 40 random modulatory groups (see Methods Section 1.10). (**b**) Example target, fit, and output (grey, red dashed, and orange curves, respectively) producing the 50th-smallest output error over 100 randomly generated combinations (see Methods Section 1.10) of l library elements using $l = 2$, $l = 4$, $l = 8$, and $l = 16$. (**c**) Fit error versus output error for 100 randomly generated combinations of l library elements for $l = 1,\ldots, 20$. We show the identity line in grey. Each point represents the 50th-smallest error between the output and the fit across 100 novel target movements. (**d**)Median errors of the 100 randomly-generated combinations of l library elements versus the number of library elements.

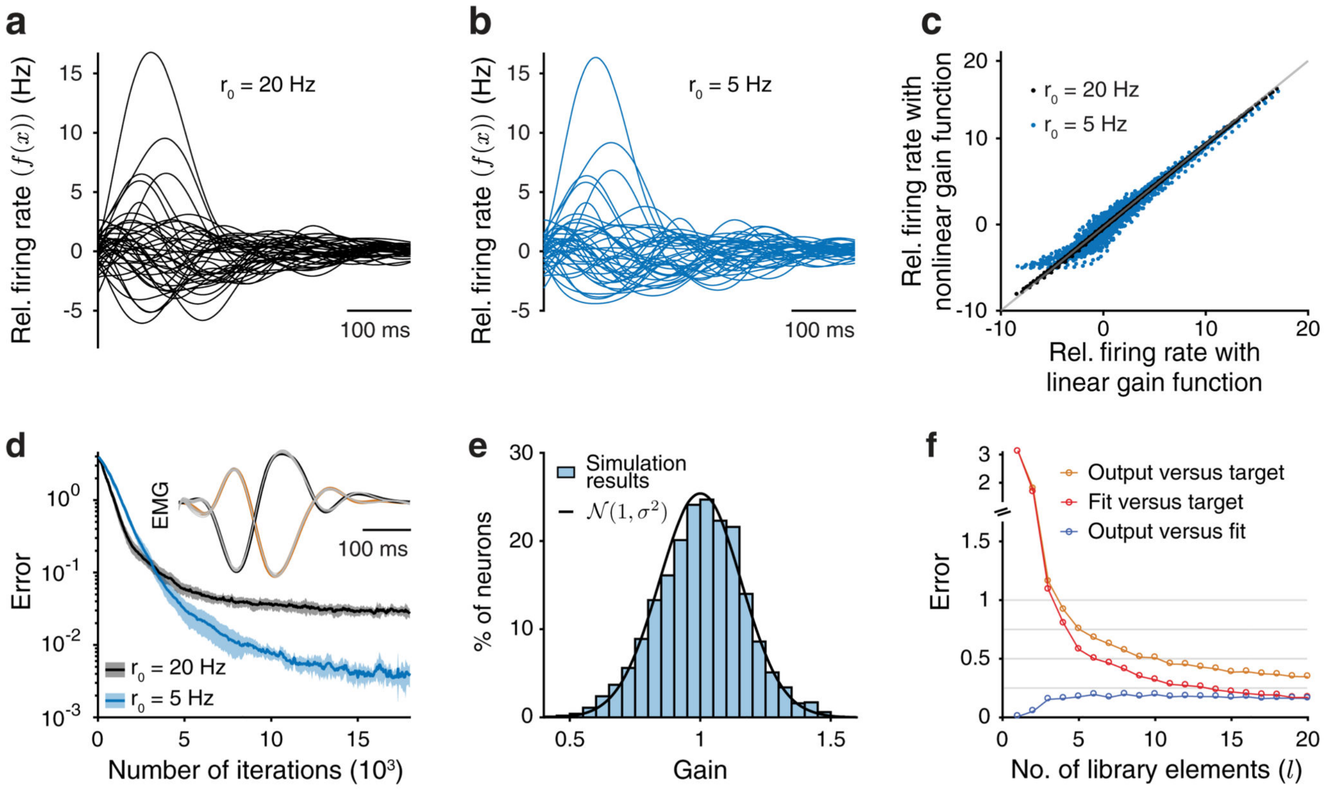

Figure 5. Examining effects of more strongly nonlinear neuronal dynamics by using a baseline rate of r_0 = 5 Hz.

(**a**) Relative firing rate of 20 excitatory and 20 inhibitory neurons in a 200-neuron network with r_0 = 20 Hz in Eqn. (2). (**b**) Relative firing rate of the same neurons as those in panel (a), but with r_0 = 5 Hz. (**c**) The dotted curves show the relative firing rates of all neurons over time when using the nonlinear gain function (see Eqn. (2)) with (black) r_0 = 20 Hz and (blue) r_0 = 5 Hz versus the relative firing rates that result from using the linear gain function $f(x_i; g_i) = g_i x_i$. We set each neuronal gain g_i to 1, and we plot the identity line in grey. (**d**) Mean error over 10 independent training sessions with r_0 = 20 Hz (black) and with r_0 = 5 Hz (blue) for the task in Fig. 1d (see Methods Section 1.10). Shading indicates one standard deviation. In the inset, we show network outputs with all gains set to 1 and the new learned gain pattern with r_0 = 5 Hz for 10 noisy initial conditions (grey curves). We show the two targets in black and orange (see Methods Section 1.10). (**e**) Histogram of gain values after training with r_0 = 5 Hz. The black curve is a Gaussian distribution with a mean of 1 and a standard deviation of $\sigma \approx 0.157$ (i.e., the distribution that we obtained with r_0 = 20 Hz in Fig. 1e). (**f**) Gain patterns as motor primitives with r_0 = 5 Hz. We generate these results in the same manner as our results in Fig. 4d, except that now we use r_0 = 5 Hz. We obtain qualitatively similar results to our observations for the baseline rate r_0 = 20 Hz.

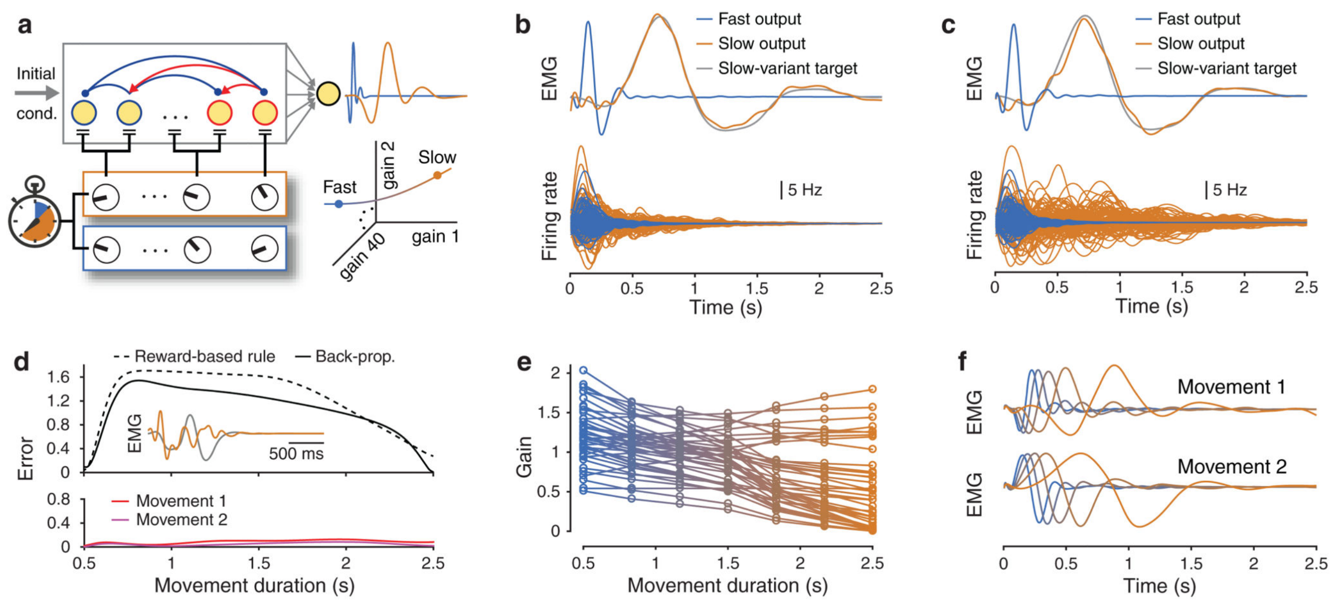

Figure 6. Gain modulation can control movement speed.

(**a**) Schematic of gain patterns for fast (0.5 s) and slow (2.5 s) movement variants. (Here and throughout the figure, we show the former in blue and the latter in orange.) We train a 400-neuron network using 40 random modulatory groups for all simulations (see Methods Section 1.10). (**b**) (Top) We train a network to extend its output from a fast to a slow-movement variant using our reward-based learning rule. (Bottom) Example firing rates of 50 excitatory and 50 inhibitory neurons for both fast and slow speed variants. (**c**) The same as panel (b), but now we use a back-propagation algorithm to train the neuronal gains (see Methods Section 1.10). (**d**) (Top) Interpolation between fast and slow gain patterns does not reliably generate target outputs of intermediate speeds when trained only at the fast and slow speeds. We show an example output (orange) that lasts a duration of 1.5 s and the associated target (grey). (Bottom) Linear interpolation between the fast and slow gain patterns successfully generates target outputs when trained at 5 intermediate speeds. We train 1 set of gain patterns (see panel (e)) on two target outputs associated with 2 different initial conditions (see Methods Section 1.10). (We plot these results with the same axis scale as in the top panel.) (**e**) The 7 optimized gain patterns for all 40 modulatory groups when training at 7 evenly-spaced speeds. (**f**) Both outputs when linearly interpolating at 5 evenly-spaced speeds between the fast and slow gain patterns from panel (e).

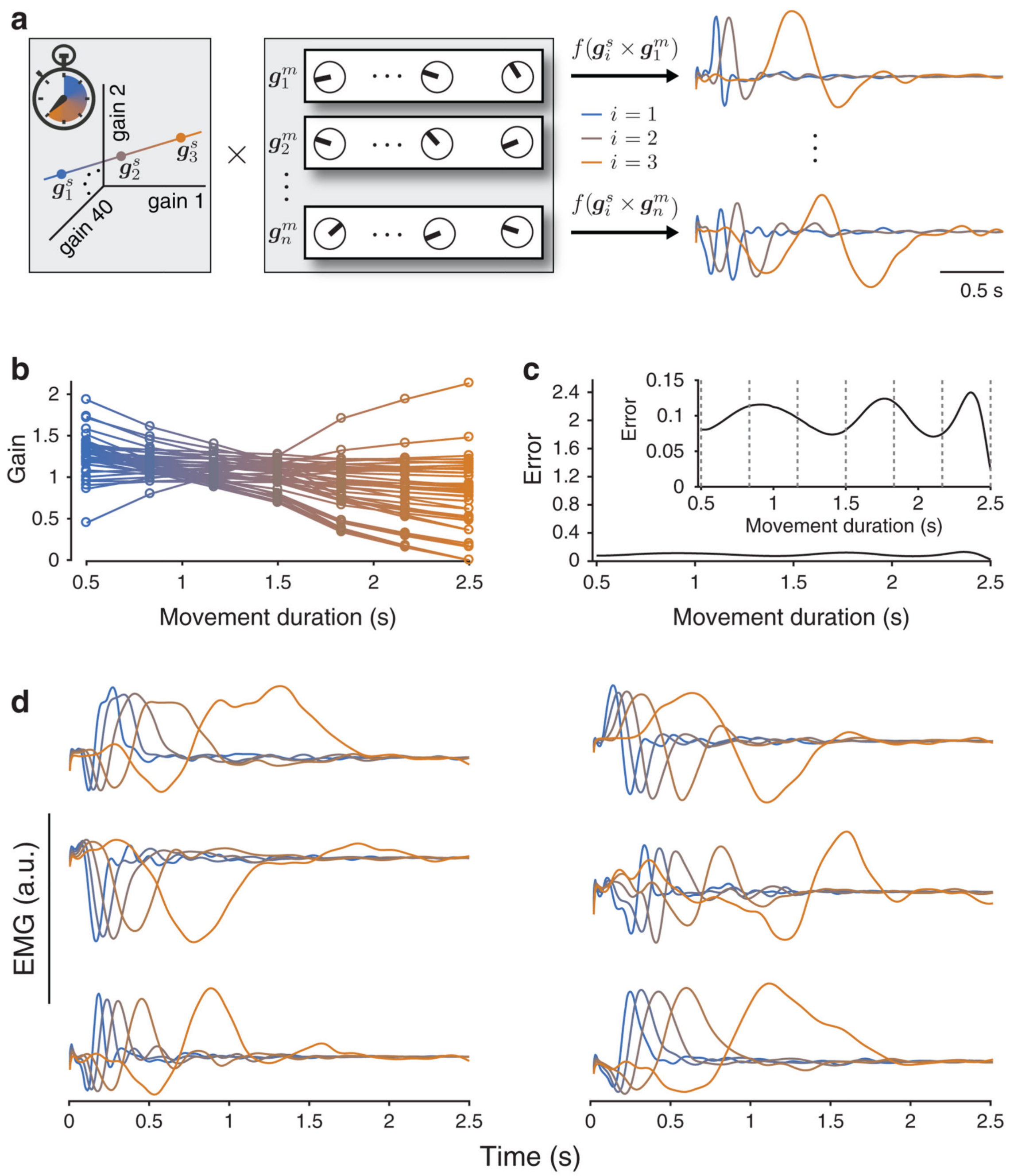

Figure 7. Joint control of movement shape and speed through gain modulation.

(**a**) One can jointly learn the gain patterns g_i^s for (left box) movement speed and g_j^m for (right box) movement shape so that the product of two such gain patterns produces a desired movement at a desired speed. In the rightmost panel, we show example outputs for two movement shapes at 3 interpolated speeds between the fast and slow gain patterns. (See the main text.) (**b**) We show the 7 optimized gain patterns for controlling movement speed (i.e., g_i^s for $i \in \{1,\ldots,7\}$ from panel (a)) for the 40 modulatory groups when training on 10 different movement shapes. (**c**) We plot the mean error over all 10 movements when linearly interpolating between the fast and slow gain patterns for controlling movement speed from panel (b). We use the same vertical axis scale as in Fig. 6d. In the inset, we plot the same data using a different vertical axis scale. The vertical dashed lines identify the 7 movement durations that we use for training. (**d**) Outputs at 5 interpolated speeds between the fast and slow gain patterns for 6 of the 10 movements. (For each simulation, we train a 400-neuron network using 40 random modulatory groups (see Methods Section 1.10).)

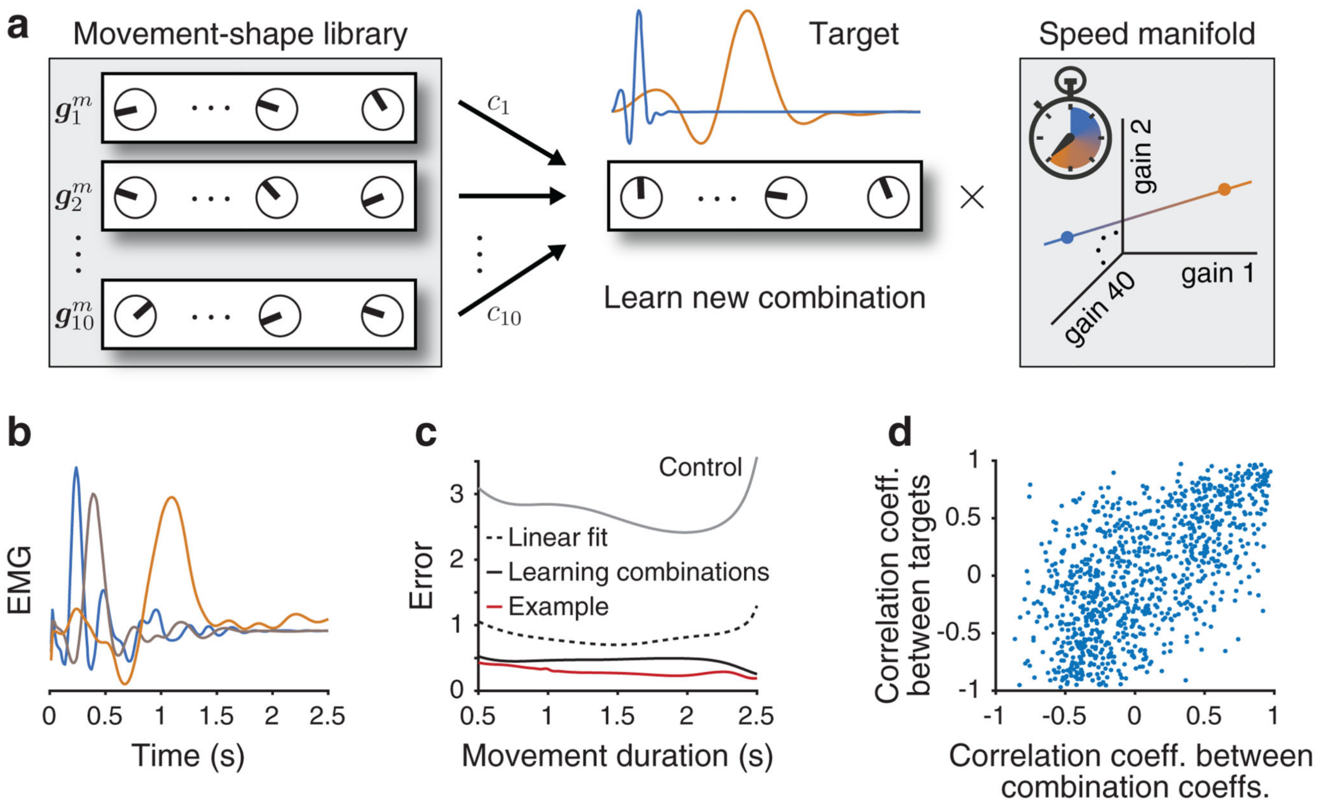

Figure 8. Learning gain-pattern primitives to control movement shape and speed.

(**a**) We are able to learn to combine (left) previously acquired gain patterns for movement shapes to generate (centre) a new target movement at both fast and slow speeds simultaneously using (right) a fixed manifold in neuronal gain space for controlling movement speed (see Methods Section 1.10). (**b**) We plot the output, at 3 different speeds, that produces the 50th-smallest error (across all 100 target movements) between the output and the target when summing errors at both fast and slow speeds. (**c**) Mean network output error across all 100 target movements for all durations when learning to combine gain patterns (black solid curve). We plot the error for the output from panel (b) in red. As a control, we plot the mean error over all target movements when dissociating the learned gain patterns from their target movement by permuting (uniformly at random) the target movements (see the grey curve). We also plot the mean error over all target movements when combining gain patterns using a least-squares fit of the 10 learned movement shapes to the target (black dashed curve) (see Methods Section 1.10). (For each example, to generate outputs of a specific duration, we linearly interpolate between the fast and slow gain patterns.) (**d**) We plot the Pearson correlation coefficient between each pair of target movements versus the Pearson correlation coefficient between the corresponding pair of learned combination coefficients$c_1, \ldots, c_{10}$.