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

One-sentence summary: Modulation of single-neuron excitability can flexibly control neuronal activity in recurrent cortical network models with fixed connectivity.

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

Motor cortex (M1) exhibits a rich repertoire of activities to support the generation of complex movements. Recent network models capture many qualitative aspects of M1 dynamics, but they can generate only a few distinct movements (all of the same duration). We demonstrate that simple modulation of neuronal input–output gains in recurrent neuronal network models with fixed connectivity can dramatically reorganize neuronal activity and consequently downstream muscle outputs. We show that 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, novel movements can be assembled from previously-learned primitives and we 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.

Motor systems continually adapt and refine voluntary movements by flexibly controlling neuronal activity in motor-related brain areas [1]. To understand how a cortical network can efficiently generate a large variety of outputs, we begin with an existing model of motor cortex that incorporates strong excitatory recurrent interactions that are stabilized by feedback inhibition [2]. For appropriate initial conditions (see Section 1.1), this model produces naturalistic activity transients (see Section 1) that resemble M1 recordings [3], and the population activity is rich enough to enable the generation of complex movements through linear readouts (see Fig. 1). However, it is unclear how the static architecture of such models allows variations in both output trajectories and their speed—e.g., to switch downstream muscle activity from one reaching movement to another (see Fig. 1A).

A possible mechanism for effectively switching network activity 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 [4–6]. Indeed, neuromodulation in M1 can cause such changes in neuronal responsiveness [7, 8], and gain modulation of motor neurons has been linked experimentally to optimization of muscular control [9, 10]. In our model, we emulate neuromodulation by including a set of modulatory afferents that directly control the input–output gain of each neuron (see Fig. 1B and Section 1). We find that uniformly increasing the gain of all neurons increases both the frequency and amplitude of the neuronal dynamics (see Fig. 1C), and the same network can produce different, yet predictable (see Section 1.7), activity trajectories.

To allow more precise control of network 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 learning rule (see Section 1.8). Our rule, which acts on the modulatory pathway of the model but is similar to proposed synaptic plasticity rules for reward-based learning [11–14], uses a global scalar signal of recent performance to iteratively evaluate and adjust each neuron's gain while network initial condition and architecture remain fixed. Starting with a network that produces an initial movement with all gains set to 1 (see 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 iterations (see Fig. 1D and 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 correlated gain patterns (see Fig. S1B). Counterintuitively, the recurrent neuronal dynamics change only slightly even though the muscle output is altered substantially (see Figs. S1C,F). Once the target is learned, the same initial condition can produce either of two distinct muscle outputs, depending on the applied gain pattern. The outputs are similarly robust with respect to noisy initial conditions for each gain pattern (see Fig. 1F), and we achieve similar learning performance (i.e., error reduction) using alternative, commonly used models of movement generation that rely on additional preparatory periods [2], or altogether different, 'chaotic' dynamics [15, 16] (see Fig. S1E and Section 1.10). Notably, in all of these models, 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 synaptic architecture or network initial condition.

Individually modulating the gain of every neuron in motor cortex is likely unrealistic. In line with the existence of diffuse (i.e., non-neuron specific) neuromodulatory projections to M1 [7, 17, 18], we cluster neurons into groups so that units within a group are modulated identically (see Fig. 2A and Section 1.11). We find that such coarse-grained modulation gives similar performance to neuron-specific control for as few as 20 randomly-formed groups in a network of 200 neurons (see Fig. 2B and Fig. S3A). For a specified number of groups, performance can be improved 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. 2B and Section 1.11). Importantly, there exist specialized groupings that perform similarly across multiple different movements (see Fig. 2C and Figs. S3B,C). Such specialized groupings acquired from learning one set of movements also perform well on novel movements (see Fig. S3D).

Notably, even with random groupings, network size hardly affects learning performance for a single readout (see Fig. 2D). Performance depends on the number of groups and not 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 more independently modulated groups (see

Fig. 2E). Finally, smaller networks typically learn faster (see the bottom panel of Fig. 2E), but they ultimately exhibit poorer performance, indicating that there is a trade-off between network size, number of groups, and task complexity (i.e., the number of readout units).

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 would be much more efficient if new gain patterns could be 'intuited' as combinations of previously acquired primitives [4, 19]. To test if this is possible in our model, we first approximate a novel target movement as a (convex) combination of existing movements. (We term this 'fit' in Fig. 3; see Section 1.12.) We then use the same combination of the associated library of gain patterns to construct a new gain pattern (see Fig. 3A). Surprisingly, the resulting network output closely resembles the target movement (see Fig. 3B). One can understand this mathematically using power series expansions of the solution of the linearized neuronal dynamics (see Section 1.9). 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. 3B–D, Fig. S4, and Fig. S5). Although the idea of using motor primitives to facilitate rapid acquisition of new movements is well-established [19, 20], our model proposes the first (to our knowledge) circuit-level mechanism for achieving this objective. In addition to neuromodulatory systems [7, 8, 10], the cerebellum is a natural candidate structure to coordinate such motor primitives [20], as it is known to project to M1 and to play a critical role in error-based motor learning [20, 21].

Thus far, we have demonstrated that simple (even coarse, group-based) gain modulation enables control of muscle activity over a 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 results [22, 23], we investigate if gain changes are able to control the speed of an intended movement (see Fig. 4A and Section 1.13). We begin with a network of 400 neurons (with 40 random modulatory groups) that generates muscle activity lasting approximately 0.5 s (as in Figs. 2 and 3). Our learning rule can successfully train the network to generate a slower variant, lasting 5 times longer (see the top panel of Fig. 4B and Fig. S6A) than the original movement (see Section 1.13). 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 (somewhat less biologically plausible) regularized back-propagation algorithm to train the neuronal gains (see Section 1.13). Following training, the slow variants are learned successfully (see Fig. 4C) and are less sensitive to the same noisy initial conditions (see Fig. S6G). The neuronal dynamics oscillate transiently, with a substantially lower frequency than either the fast variants or the slow variants trained by our local learning rule (compare the bottom panels of Fig. 4C and Fig. 4B). We can also find a single gain pattern that, instead of slowing down one movement, slows down up to approximately five distinct movements (associated with five orthogonal initial conditions) by a factor of 5 (see Figs. S6H–J). Thus, the temporal scale of transient neuronal activity can be extended several-fold through specific changes in neuronal gains.

Following training on a slow and a fast variant of the same movement (see above), we find that naively interpolating between the two gain patterns does not yield the same movement at intermediate speeds (see Fig. S7A), consistent with human subjects being unable to consistently apply learned movements at novel speeds [24]. Thus, 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 to yield gain patterns for fast and slow variants 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 ranging from 0.5 s to 2.5 s) by adjusting both the readout weights and gain patterns for the fast and slow variants (see Figs. S7B,D and Section 1.13). Linear interpolation between the fast and slow gain patterns now generates smooth speed control of both movements (see Figs. 4D,E). In other words, to control movement speed, we learn a ‘manifold’ [25] in neuronal gain space that is delimited by the fast and the slow gain patterns (see Fig. 4A; bottom right).

Thus far, we have shown that gain modulation can affect either the shape or the speed of a movement. Flexible independent control of both the shape and speed of a movement (i.e., jointly) necessitates separate representations of space and time in the gain patterns. A relatively simple possibility is to find a single manifold in neuronal gain space for speed control and combine it with gain patterns associated with different movement shapes. Biologically, this may be achievable using separate modulatory systems. We achieve such separation by simultaneously training one manifold for speed control and 10 gain patterns for 10 different movement shapes in a model in which the movements are encoded by the product of shape-specific and speed-specific gain patterns (see Fig. 4F, Figs. S7E,F, and Section 1.13). We thereby obtain separate families of gain patterns for movement shape and speed that independently control movements in space and time.

Our results support the view that knowing only the structure of neuronal networks does not suffice to explain their dynamics [26, 27]. In line with known physiological effects [6, 8, 26, 28], we have shown that relatively subtle changes in neuronal excitability in cortical circuits can have dramatic effects on ensuing muscular activity, suggesting the possibility that gain modulation is a central part of neuronal motor control. Gain modulation may occur primarily via neuromodulators [8, 10], as we suggest in this paper, but it may also arise from changes in the balance of excitatory and inhibitory inputs to cortical neurons (for example, through inputs from the cerebellum) [29]. Indeed, in real cortical circuits, changes in neuronal dynamics will likely stem from changes in both inputs [30] and modulatory states [31].

In traditional theories of learning, synaptic modifications occur directly in the circuit whose activity expresses the (motor) memory [18, 32], which would result in altered dynamics in these networks even during periods of idle behaviour, thus providing experimentally accessible signatures of learning. In contrast, our work predicts that synaptic modifications take place further upstream—for example, in the input synapses to the presumed neuromodulatory neurons [33]. Therefore, once the trained modulatory input is removed, neuronal activity would not exhibit any sign of learning other than during epochs of movement generation. Consequently, elucidating the neural substrate of motor learning may necessitate recording from a potentially broader set of brain areas than those circuits whose activity correlates directly with movement dynamics.

References

1. D. A. Rosenbaum, *Human Motor Control.* Cambridge, USA: Academic Press, 2009.

2. G. Hennequin, T. P. Vogels, and W. Gerstner, "Optimal control of transient dynamics in balanced networks supports generation of complex movements," *Neuron*, vol. 82, no. 6, pp. 1394–1406, 2014.

3. M. M. Churchland, J. P. Cunningham, M. T. Kaufman, J. D. Foster, P. Nuyujukian, S. I. Ryu, and K. V. Shenoy, "Neural population dynamics during reaching," *Nature*, vol. 487, no. 7405, pp. 1–8, 2012.

4. C. D. Swinehart, K. Bouchard, P. Partensky, and L. F. Abbott, "Control of network activity through neuronal response modulation," *Neurocomputing*, vol. 58, pp. 327–335, 2004.

5. J. Zhang and L. F. Abbott, "Gain modulation of recurrent networks," *Neurocomputing*, vol. 32, pp. 623–628, 2000.

6. E. Marder, "Neuromodulation of neuronal circuits: Back to the future," *Neuron*, vol. 76, no. 1, pp. 1–11, 2012.

7. K. Molina-Luna, A. Pekanovic, S. Rohrich, B. Hertler, M. Schubring-Giese, M. S. Rioult-Pedotti, and A. R. Luft, "Dopamine in motor cortex is necessary for skill learning and synaptic plasticity," *PloS One*, vol. 4, no. 9, p. e7082, 2009.

8. K. Thurley, W. Senn, and H.-R. Lüscher, "Dopamine increases the gain of the input–output response of rat prefrontal pyramidal neurons," *Journal of Neurophysiology*, vol. 99, no. 6, pp. 2985–2997, 2008.

9. M. Vestergaard and R. W. Berg, "Divisive gain modulation of motoneurons by inhibition optimizes muscular control," *Journal of Neuroscience*, vol. 35, no. 8, pp. 3711–3723, 2015.

10. K. Wei, J. I. Glaser, L. Deng, C. K. Thompson, I. H. Stevenson, Q. Wang, T. G. Hornby, C. J. Heckman, and K. P. Körding, "Serotonin affects movement gain control in the spinal cord," *Journal of Neuroscience*, vol. 34, no. 38, pp. 12690–12700, 2014.

11. I. R. Fiete and H. S. Seung, "Gradient learning in spiking neural networks by dynamic perturbation of conductances," *Physical Review Letters*, vol. 97, no. 4, p. 048104, 2006.

12. R. Legenstein, S. M. Chase, A. B. Schwartz, and W. Maass, "A reward-modulated Hebbian learning rule can explain experimentally observed network reorganization in a brain control task," *Journal of Neuroscience*, vol. 30, no. 25, pp. 8400–8410, 2010.

13. G. M. Hoerzer, R. Legenstein, and W. Maass, "Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning," *Cerebral Cortex*, vol. 24, no. 3, pp. 677–690, 2014.

14. T. Miconi, "Biologically plausible learning in recurrent neural networks for flexible decision tasks," *eLife*, vol. 6, p. e20899, 2017.

15. D. Sussillo and L. F. Abbott, "Generating coherent patterns of activity from chaotic neural networks," *Neuron*, vol. 63, no. 4, pp. 544–557, 2009.

16. H. Sompolinsky, A. Crisanti, and H. J. Sommers, "Chaos in random neural networks," *Physical Review Letters*, vol. 61, no. 3, pp. 259–262, 1988.

17. G. W. Huntley, J. H. Morrison, A. Prikhozhan, and S. C. Sealfon, "Localization of multiple dopamine receptor subtype mRNAs in human and monkey motor cortex and striatum," *Molecular Brain Research*, vol. 15, no. 3-4, pp. 181–188, 1992.

18. J. A. Hosp, A. Pekanovic, M. S. Rioult-Pedotti, and A. R. Luft, "Dopaminergic projections from midbrain to primary motor cortex mediate motor skill learning," *Journal of Neuroscience*, vol. 31, no. 7, pp. 2481–2487, 2011.

19. S. F. Giszter, "Motor primitives—New data and future questions," *Current Opinion in Neurobiology*, vol. 33, pp. 156–165, 2015.

20. K. A. Thoroughman and R. Shadmehr, “Learning of action through adaptive combination of motor primitives,” *Nature*, vol. 407, no. 6805, pp. 742–747, 2000.

21. D. A. Spampinato, H. J. Block, and P. A. Celnik, “Cerebellar–M1 connectivity changes associated with motor learning are somatotopic specific,” *Journal of Neuroscience*, vol. 37, no. 9, pp. 2377–2386, 2017.

22. J. Wang, D. Narain, E. A. Hosseini, and M. Jazayeri, “Flexible timing by temporal scaling of cortical responses,” *Nature Neuroscience*, vol. 21, no. 1, pp. 102–110, 2018.

23. S. Soares, B. V. Atallah, and J. J. Paton, “Midbrain dopamine neurons control judgment of time,” *Science*, vol. 354, no. 6317, pp. 1273–1277, 2016.

24. N. F. Hardy, V. Goudar, J. L. Romero-Sosa, and D. V. Bounomano, “A model of temporal scaling correctly predicts that Weber’s law is speed-dependent,” *bioRxiv*, p. 159590, 2017.

25. J. A. Gallego, M. G. Perich, L. E. Miller, and S. A. Solla, “Neural manifolds for the control of movement,” *Neuron*, vol. 94, no. 5, pp. 978–984, 2017.

26. C. I. Bargmann, “Beyond the connectome: How neuromodulators shape neural circuits,” *BioEssays*, vol. 34, no. 6, pp. 458–465, 2012.

27. D. S. Bassett and O. Sporns, “Network neuroscience,” *Nature Neuroscience*, vol. 20, no. 3, pp. 353–364, 2017.

28. H. Kida and D. Mitsushima, “Mechanisms of motor learning mediated by synaptic plasticity in rat primary motor cortex,” *Neuroscience Research*, pp. 1–5, 2017.

29. F. S. Chance, L. F. Abbott, and A. D. Reyes, “Gain modulation from background synaptic input,” *Neuron*, vol. 35, no. 4, pp. 773–782, 2002.

30. G. F. Elsayed, A. H. Lara, M. T. Kaufman, M. M. Churchland, and J. P. Cunningham, “Reorganization between preparatory and movement population responses in motor cortex,” *Nature Communications*, vol. 7, p. 13239, 2016.

31. V. R. Athalye, F. J. Santos, J. M. Carmena, and R. M. Costa, “Evidence for a neural law of effect,” *Science*, vol. 359, no. 6379, pp. 1024–1029, 2018.

32. L. F. Abbott and S. B. Nelson, “Synaptic plasticity: Taming the beast,” *Nature Neuroscience*, vol. 3, pp. 1178–1183, 2000.

33. A. R. O. Martins and R. C. Froemke, “Coordinated forms of noradrenergic plasticity in the locus coeruleus and primary auditory cortex,” *Nature Neuroscience*, vol. 18, no. 10, pp. 1483–1492, 2015.

34. K. Rajan, L. F. Abbott, and H. Sompolinsky, “Stimulus-dependent suppression of chaos in recurrent neural networks,” *Physical Review E*, vol. 82, no. 1, p. 011903, 2010.

35. J. C. Kao, P. Nuyujukian, S. I. Ryu, M. M. Churchland, J. P. Cunningham, and K. V. Shenoy, “Single-trail dynamics of motor cortex and their applications to brain-machine interfaces,” *Nature Communications*, vol. 6, pp. 1–12, 2015.

36. G. Teschl, *Ordinary Differential Equations and Dynamical Systems*. American Mathematical Society, 2012.

37. N. Frémaux and W. Gerstner, “Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules,” *Frontiers in Neural Circuits*, vol. 9, p. 85, 2016.

38. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

Acknowledgements: We thank the members of the Vogels lab (particularly Everton J. Agnes, Rui P. Costa, and William F. Podlaski) and Yayoi Teramoto Kimura for creating the monkey illustration. We also thank Omri Barak, Tim E. Behrens, and Rafal Bogacz for their helpful comments. **Funding:** Our work was supported by grants from the Wellcome Trust (TPV and JPS WT100000, GH 202111/Z/16/Z) and the Engineering and Physical Sciences Research Council (JPS). **Authors contributions:** JPS, GH, and TPV conceived the study. JPS performed simulations for Figs. 1–3 and Figs. S1–S5, JPS and GH performed simulations for Fig. 4 and Figs. S6–S7. 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 revised the final manuscript. **Competing interests:** The authors declare that no competing interests exist. **Data and materials availability:** Code or other materials will be made available upon reasonable request. Requests should be addressed to JPS (`jake.stroud@cncb.ox.ac.uk`).

1 Supplementary Methods

1.1 Network dynamics

Following Ref. [2], we use recurrent rate networks of $N = 2M$ neurons (M excitatory and M inhibitory) whose state $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))^\top$ evolves according to the dynamical system

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

from some initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$. In Eqn. (1), $f(\boldsymbol{x}; \boldsymbol{g})$ denotes the element-wise application of the static scalar function f to the neuronal activity vector $\boldsymbol{x}$. We choose the initial state $\boldsymbol{x}_0$ among the 'most observable' modes of the system (i.e., those that elicit strong, temporally-rich activity transients [2]). Specifically, we first linearize the dynamics around its unique fixed point $\boldsymbol{x} = 0$ using unit gains (i.e., all $g_i = 1$), and computed the observability Gramian (a symmetric positive-definite matrix $\boldsymbol{Q}$) of the linearized system [2]. The most observable modes (i.e., the initial conditions that evoke the largest transients) are the top eigenvectors of $\boldsymbol{Q}$. We detail the choice of synaptic weight matrix $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ in Section 1.3.

We do not explicitly model dynamics prior to movement execution; all of our simulations begin at the time of movement onset [2, 3]. In keeping with [2], we set the single-neuron time constant to be $\tau = 200$ ms, and the gain function f, which governs the transformation of neuronal activity $\boldsymbol{x}$ into firing rates relative to a baseline rate r_0, is

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

where g_i is the slope of the function f at baseline rate r_0 and thus controls the input–output sensitivity of neuron i [34]. We use a baseline rate of $r_0 = 20$ Hz and a maximum firing rate of $r_{\max} = 100$ Hz, consistent with observations [3]. (See Fig. 1C, where we plot the gain function shifted up by 20 Hz.) With this setup, the majority of the neuronal dynamics operate within the linear part of the nonlinear gain function f (i.e., the neuronal dynamics are similar to the case of using a linear gain function (see Fig. S2C)) — which is consistent with experimental observations [35]. However, by reducing r_0 so that it is closer to 0, which leads to more neuronal activities near the lower saturation regime of the nonlinear gain function f (i.e., more nonlinear behaviour), we obtain qualitatively similar results to those that we presented in the main manuscript. We demonstrate several of our main results using $r_0 = 5$ Hz in Figs. S2 and S5.

1.2 Biophysical interpretation of Eqn. (1)

In Section 1.1, we described how neuronal activity can be modelled 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 models the neuronal activity as

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

for the same initial condition x_0 that we described in Section 1.1, where $h_i = -r_0 \sum_j W_{ij}$ and

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

where $r_{\max}$ is the maximum firing rate.

1.3 Construction of the network architecture

Prior to optimization, we generate synaptic weight matrices $\boldsymbol{W}$ as detailed in Ref. [2]. 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 and strong weights with nonzero elements set to $w_0/\sqrt{N}$ (excitatory) and $-\gamma w_0/\sqrt{N}$ (inhibitory), where $w_0^2 = 2\rho^2/(p(1-p)(1+\gamma^2))$ and the connection probability between each two neurons is $p = 0.1$. This construction results in $\boldsymbol{W}$ having a circular eigenvalue spectrum of radius ρ, set to $\rho = 10$, leading to linear instabilities before stability optimization (see below). As in Ref. [2], 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 [2], 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 $\boldsymbol{W}$) [2]. Briefly, inhibitory weights are iteratively updated 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 ($\gamma = 3$) of mean inhibitory to mean excitatory weights. Third, we restrict the density of inhibitory connections to be less than or equal to 0.4 to maintain sparse connectivity. This constrained gradient descent usually converges within a few hundred iterations. As was also observed in Ref. [2], the SA typically decreases during optimization from 10 to about 0.15. For additional details, see the supplemental information of Ref. [2].

As a proof of principle, we also construct a 'chaotic' variant of our network model (see Fig. S1E). These networks are chaotic in the sense that the neuronal dynamics Eqn. (1) give rise to a positive maximal Lyapunov exponent [16]. We use a synaptic weight matrix $\boldsymbol{W}$, as described above prior to optimization, but with $\gamma = 1$ and $\rho = 1.5$ (as in Ref. [15]). We set $\tau = 20$ ms, and we choose the initial condition for each neuron's activity from a uniform distribution between -10 and 10 Hz.

1.4 Creating target muscle activity

We generate target muscle activities of duration $T = 500$ ms (fast movements) and $T = 2500$ ms (slow movements). In each case, we draw muscle activity from a Gaussian process with a covariance function $K \in [0:T] \times [0:T] \to \mathbb{R}^+$ 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 EMG data during reaching [3]. Specifically,

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

with $E(t) = t\exp(-t^2/4)$. We set $\sigma = 125$ ms and $\ell = 30$ ms for fast movements and $\sigma = 624$ ms and $\ell = 150$ ms for slow movements. 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 fast movements and 200 Hz for slow movements.

1.5 Network output

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

$$z(t) = \boldsymbol{m}^\top f(\boldsymbol{x}^E(t); \boldsymbol{g}^E) + b\,, \tag{6}$$

where $\boldsymbol{m}\,, \boldsymbol{x}^E(t)\,, \boldsymbol{g}^E \in \mathbb{R}^M$ and $\boldsymbol{x}^E(t)$ is the excitatory neuronal activity. To ensure that the network generates realistic muscle activity (see 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 using least-squares regression. To ameliorate any issues of overfitting, we use 100 noisy trials, in which we add Gaussian white noise to the network initial condition $\boldsymbol{x}_0$ for each trial with a signal-to-noise ratio of 30 dB [2]. Subsequently, the readout weights remain fixed throughout training of the neuronal gains. Additionally, see our simulation details for each figure.

1.6 Network output error

We compute the error ϵ between the network output $\boldsymbol{z} \in \mathbb{R}^T$ and the target $\boldsymbol{y} \in \mathbb{R}^T$ by calculating

$$\epsilon = 1 - R^2 = \frac{\sum_{t=1}^{T}(z(t) - y(t))^2}{\sum_{t=1}^{T}(y(t) - \bar{y})^2}\,, \tag{7}$$

where $\bar{y} = \frac{1}{T}\sum_{t=1}^{T} y(t)$ and R^2 is the commonly used coefficient of determination (which is often called simply ‘R-squared’). Therefore, an error of $\epsilon = 1$ implies that the performance is as bad as if the output $\boldsymbol{z}$ is 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 Analysis of the effects of identically changing the gain of all neurons

To examine the effects of gain modulation on neuronal dynamics when identically changing all neuronal gains (i.e., $g_i = g$ for all i), we construct a Taylor expansion of $f(x_i; g_i)$ from Eqn. (2). By keeping only leading-order terms, we obtain $f(x_i; g) \approx g x_i$, and substituting this expression into Eqn. (1) yields $\tau\dot{\boldsymbol{x}} = (g\boldsymbol{W} - \boldsymbol{I}) \cdot \boldsymbol{x} = \boldsymbol{A} \cdot \boldsymbol{x}$, where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{A} = g\boldsymbol{W} - \boldsymbol{I}$. Empirically, we find this linear approximation to be valid in a large basin of attraction around the equilibrium.

Changing the gain from g to g' multiplies the imaginary part of the spectrum of $\boldsymbol{A}$ by the factor g'/g. (Subtracting the identity matrix does not affect the imaginary part of the spectrum of $\boldsymbol{A}$.) This, in turn, multiplies the frequency of the associated solution of the linearized dynamics of $\boldsymbol{x}(t)$ by the factor of g'/g.

A change in gain also causes changes in the real parts of the eigenvalues of $\boldsymbol{A}$. Specifically, increasing the gain causes the real parts of all but one of the eigenvalues of $g\boldsymbol{W}$ to increase (i.e., the eigenvalues of $\boldsymbol{A}$ get closer to the imaginary axis), generally causing a slower decay of activity towards the equilibrium [36]. The real part of the remaining eigenvalue, which is associated with the eigenvector $(1,1,\ldots,1)^T/\sqrt{N}$ (see Ref. [2]), becomes more negative with increasing gain, resulting in faster decay of the neuronal dynamics (however, this effect is small compared with the slowing of the decay due to the changes of the other real parts of the eigenvalues mentioned above).

1.8 A learning rule for neuronal input–output gains

We devise a reward-based 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 [11, 12, 14]. 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:

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

where

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

where $\epsilon(t_n)$ represents the output error at iteration t_n (see 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 reward signal is $R(t_0) = 1$, and the other initial conditions are $\bar{\epsilon}(t_0) = \epsilon(t_0)$ (where $\epsilon(t_0)$ 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 $\bar{\epsilon}$ as low-pass-filtered gains and errors, respectively, over recent iterations with a history controlled by the decay rate α [14]. We use these parameter values in all of our simulations. 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. [13].

The learning rule (8) is similar to the reward-modulated 'exploratory Hebbian' (EH) synaptic plasticity rule [12–14]. However, we investigate gain learning in which there are changes in neuronal gains (i.e., the responsiveness of neurons) inside a recurrent network rather than on the synaptic readout weights (as was explored in Refs. [12, 13]). Additionally, our reward signal R does not provide information on 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. One can view the reward signal as an abstract model for phasic output of dopaminergic systems in the brain [7, 17, 18, 37].

We update the gains as follows. 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 $\epsilon(t_1)$) from the output, and we subsequently calculate the reward $R(t_1)$ and the quantities $\bar{\epsilon}(t_1)$ and $\bar{g}(t_1)$ using Eqn. (9). We then repeat this process for all subsequent iterations.

One can also adapt our learning rule so that learning ceases when the error $\epsilon(t_n)$ 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 reward term R multiplies ξ_i, together with changing the sgn function in Eqn. (9) to the $\tanh$ function. This yields the following learning rule:

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

where

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

and $\eta = 50,000$ controls the slope of the $tanh$ function at 0 (i.e., when the low-pass-filtered error $\bar{\epsilon}(t_n)$ matches the current error $\epsilon(t_n)$). Learning now stops when $\bar{\epsilon}(t_{n-1}) = \epsilon(t_n)$; see the orange curve in Fig. S1E. We achieve a qualitatively similar learning performance by using Eqns. (10) and (11) instead of Eqns. (8) and (9), respectively. (Compare the orange and red curves in Fig. S1E.)

1.9 Analysis of linear combinations of gain patterns and their associated solutions

In Fig. 3, we illustrated that there is a consistent mapping between learned gain patterns and their outputs. Specifically, we illustrated that for a library of k gain patterns ($\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$), a convex combination $c_1 f(\boldsymbol{g}_1) + \ldots + c_k f(\boldsymbol{g}_k)$ (so $c_j \geq 0$ for all j and $\sum_{j=1}^{k} c_j = 1$) of their corresponding outputs approximates the output $f(c_1\boldsymbol{g}_1 + \ldots + c_k\boldsymbol{g}_k)$ obtained using the gain patterns combined in the same way (see Fig. 3). Here, the subscript index j denotes the library element j and is not a neuron index. We now provide some mathematical understanding of this phenomenon by studying linearized solutions of the neuronal dynamics. Because the readout unit is a linear combination of the neuronal dynamics, it is sufficient to study convex combinations of internal neuronal activity $\boldsymbol{x}(t)$ directly, rather than convex combinations of linear readout trajectories.

For a convex combination (i.e., a weighted mean) of k vectors or matrices $\boldsymbol{\phi}$ with weights c_j, it is convenient to use the following notation:

$$\mathcal{C}\left[\tilde{\boldsymbol{\phi}}\right] = \sum_{j=1}^{k} c_j \boldsymbol{\phi}_j\,, \tag{12}$$

where the tilde in the square brackets is a reminder that we are summing over the index of the associated library terms. For a given gain pattern $\boldsymbol{G}_j \in \mathbb{R}^{N\times N}$ (where the neuronal gains are elements along the diagonal of $\boldsymbol{G}_j$ and all other elements are 0, and the index j denotes library element j), the solution $\boldsymbol{x}_j \in \mathbb{R}^N$ of the linearized dynamics of Eqn. (1) (i.e., we linearize the gain function f) is given by

$$\boldsymbol{x}_j(t) = e^{\frac{t}{\tau}(\boldsymbol{W}\boldsymbol{G}_j - \boldsymbol{I})}\boldsymbol{x}_0\,, \tag{13}$$

under the assumption that there are N distinct eigenvectors for the matrix $\boldsymbol{W}\boldsymbol{G}_j - \boldsymbol{I}$ and that we are away from any bifurcations. Let

$$\boldsymbol{u}(t) = e^{\frac{t}{\tau}\left[\boldsymbol{W}\mathcal{C}[\tilde{\boldsymbol{G}}] - \boldsymbol{I}\right]}\boldsymbol{x}_0 \tag{14}$$

denote the neuronal activity that results from a convex combination $\mathcal{C}\left[\tilde{\boldsymbol{G}}\right]$ of gain patterns. We need to show that $\boldsymbol{u}(t)$ is approximately the same as the convex combination of the individual

407 neuronal dynamics $\boldsymbol{x}_j(t)$ with the same coefficients c_j. That is, we need to show that the difference

$$\boldsymbol{\Delta}(t) = \boldsymbol{u}(t) - \mathcal{C}\left[\tilde{\boldsymbol{x}}(t)\right] \tag{15}$$

is small with respect to the magnitude of the neuronal activity. We first note that $\frac{d\boldsymbol{\Delta}}{dt}\big|_{t=0} = \boldsymbol{0}$, which we prove as follows:

$$\begin{aligned} \frac{d}{dt}\boldsymbol{u}(t)\bigg|_{t=0} &= \frac{1}{\tau}\left(\boldsymbol{W}\mathcal{C}\left[\tilde{\boldsymbol{G}}\right] - \boldsymbol{I}\right)\boldsymbol{x}_0 \\ &= \frac{1}{\tau}\mathcal{C}\left[\boldsymbol{W}\tilde{\boldsymbol{G}} - \boldsymbol{I}\right]\boldsymbol{x}_0 \\ &= \frac{d}{dt}\mathcal{C}\left[\tilde{\boldsymbol{x}}(t)\right]\bigg|_{t=0}, \end{aligned} \tag{16}$$

where we used the fact that $\sum_{j=1}^{k} c_j = 1$ to go from the first to the second line and the matrices $\boldsymbol{W}$ and $\boldsymbol{I}$ do not depend on the gain patterns. To see whether we can also expect $\boldsymbol{\Delta}(t)$ to be small for $t > 0$, it is useful to consider the power-series expansion of the matrix exponentials on the right-hand side of Eqn. (15):

$$\mathcal{C}\left[\tilde{\boldsymbol{x}}(t)\right] = \mathcal{C}\left[\left(\sum_{m=0}^{\infty}\frac{(\boldsymbol{W}\tilde{\boldsymbol{G}} - \boldsymbol{I})^m}{m!}\right)^{\frac{t}{\tau}}\boldsymbol{x}_0\right], \tag{17}$$

$$\boldsymbol{u}(t) = \left(\sum_{m=0}^{\infty}\frac{\left(\boldsymbol{W}\mathcal{C}\left[\tilde{\boldsymbol{G}}\right] - \boldsymbol{I}\right)^m}{m!}\right)^{\frac{t}{\tau}}\boldsymbol{x}_0\,. \tag{18}$$

We observe in numerical simulations (not shown) that power-series expansions of this form are accurate descriptions of the associated neuronal dynamics up to second order in m. We therefore truncate to $m = 2$, and we evaluate the difference of Eqns. (17) and (18):

$$\boldsymbol{\Delta}(t) = \left(\frac{1}{2}\right)^{\frac{t}{\tau}}\left(\mathcal{C}\left[\left((\boldsymbol{W}\tilde{\boldsymbol{G}})^2 + \boldsymbol{I}\right)^{\frac{t}{\tau}}\right] - \left(\left(\boldsymbol{W}\mathcal{C}\left[\tilde{\boldsymbol{G}}\right]\right)^2 + \boldsymbol{I}\right)^{\frac{t}{\tau}}\right)\boldsymbol{x}_0\,. \tag{19}$$

We need to check if the right-hand side of Eqn. (19) is small compared to the neuronal dynamics (i.e., compared to Eqn. (17)). One way to check if this holds at certain times t is to substitute values of t into Eqns. (19) and (17) and calculate the ratio of the norms of these two expressions. Setting $t = \tau$ — at $t = \tau = 200$ ms, the neuronal dynamics are close having reached their maximum amplitude (see Fig. S3E) — yields

$$\frac{\left\|\boldsymbol{\Delta}(t)\,|_{t=\tau}\right\|}{\left\|\mathcal{C}\left[\tilde{\boldsymbol{x}}(t)\,|_{t=\tau}\right]\right\|} \approx \frac{\left\|\left(\mathcal{C}\left[\left(\boldsymbol{W}\tilde{\boldsymbol{G}}\right)^{2}+\boldsymbol{I}\right]-\left(\boldsymbol{W}\mathcal{C}\left[\tilde{\boldsymbol{G}}\right]\right)^{2}-\boldsymbol{I}\right)\boldsymbol{x}_0\right\|}{\left\|\left(\mathcal{C}\left[\left(\boldsymbol{W}\tilde{\boldsymbol{G}}\right)^{2}+\boldsymbol{I}\right]\right)\boldsymbol{x}_0\right\|}$$

$$= \frac{\left\|\boldsymbol{W}^{2}\left(\mathcal{C}\left[\tilde{\boldsymbol{G}}^{2}\right]-\left(\mathcal{C}\left[\tilde{\boldsymbol{G}}\right]\right)^{2}\right)\boldsymbol{x}_0\right\|}{\left\|\left(\boldsymbol{W}^{2}\mathcal{C}\left[\tilde{\boldsymbol{G}}^{2}\right]+\boldsymbol{I}\right)\boldsymbol{x}_0\right\|}. \tag{20}$$

We now study the magnitude of both the numerator and the denominator of Eqn. (20) and show that the ratio is small. Both the numerator and the denominator scale approximately in linear proportion to the norm of the product of $\boldsymbol{W}^2$ and that of $\boldsymbol{x}_0$ (the identity matrix in the denominator is small compared to $\boldsymbol{W}^2$). The main difference between the two is their dependency on the gain patterns $\boldsymbol{G}_j$. The numerator scales approximately proportionally to a 'weighted variance' of the gain patterns (specifically, with $\mathcal{C}\left[\tilde{\boldsymbol{G}}^2\right] - \left(\mathcal{C}\left[\tilde{\boldsymbol{G}}\right]\right)^2$), whereas the denominator scales approximately proportionally to a weighted mean of the squared gain patterns (i.e., ($\mathcal{C}\left[\tilde{\boldsymbol{G}}^2\right]$)). Because our learned gain patterns are typically narrowly distributed, with a mean of 1 and approximate standard deviation of 0.15 (see Fig. S4A), this ratio is small (on the order of 10^{-2}). Numerically, we confirm that the normalized error in Eqn. (20) is indeed small, which also corroborates the results of Fig. 3 of the main text. Finally, we note that although we restricted our discussion above to a linear gain function, our numerical simulations suggest that Eqn. (15) is also small for the nonlinear gain function of Eqn. (2) (see Fig. 3) that we used throughout the main text.

1.10 Simulation details for Fig. 1 and Figs. S1 and S2

We create two different electromyogram (EMG) (see Section 1.4) muscle activities (initial reach and target reach) that each last 0.5 s (see Figs. 1A,F). We use a network of $N = 200$ neurons and sample transient neuronal dynamics lasting 0.5 s following the network initial condition (see Section 1.1). We fit the readout weights over 100 trials in which we add white Gaussian noise to the network initial condition $\boldsymbol{x}_0$ (with a signal-to-noise ratio of 30 dB) using least-squares regression so that the initial network output, with all gains set to 1, approximates the initial reach (see Section

1.5). We use the same readout weights throughout all training, and we use only one readout unit for all simulations.

In Fig. 1C, we plot the dynamics of three example neurons with all gains set to 1 (black) and all gains set to 2 (blue).

For each training iteration of the neuronal gains (to approximate the target movement), we give the initial condition $\boldsymbol{x}_0$ to the network at time $t = 0$ (see Section 1.1), and we calculate the subsequent network output as described in Section 1.5. We compute the error ϵ (see Section 1.6) after each iteration, and we then update the neuronal gains according to Eqn. (8). We repeat this process for $18,000$ training iterations (which, in physical units, corresponds to 2.5 hours of training time), which is enough training time for the error to saturate (see Fig. 1D).

We run 10 independent training sessions on the same target, and we plot these results in Figs. 1D,E. For the outputs that we show in Fig. 1F, we add white Gaussian noise to the network initial condition $\boldsymbol{x}_0$ with a signal-to-noise ratio of 30 dB using one of the learned gain patterns and with all gains equal to 1. For each of the 10 learned gain patterns $\boldsymbol{g}$, we plot the change in the spectral abscissa of $\boldsymbol{W} \times \text{diag}(\boldsymbol{g})$ (i.e., the largest real part in the spectrum of $\boldsymbol{W} \times \text{diag}(\boldsymbol{g})$) in Fig. S1A. We observe an increase in the spectral abscissa after training.

In Fig. S1B, we calculate, for each neuron, the variance of the gains across the 10 training sessions, and we plot the mean variance across all neurons (see the arrow). We also plot the distribution of mean variances from a permutation test with $10,000$ independent uniformly random shuffles of gain values across neurons and training sessions. (We obtain a p-value of $p < 10^{-4}$.) This suggests that similar gain patterns occur in independent training sessions.

To generate the correlation matrices that we show in Fig. S1C, we calculate the Pearson correlation coefficient of the neuronal dynamics between all pairs of neurons in the recurrent network. Therefore, each entry in the matrix indicates the extent to which the neuronal dynamics are similar for a pair of neurons over the duration of the movement (i.e., 0.5 s). We show correlation matrices for examples in which all gains are set to 1 and for two example learned gain patterns (see our discussion above). We use the same network initial condition that we used during training, and we observe that there is not a substantial change in the correlations between the neuronal dynamics even though we obtain a dramatically different output activity.

We also studied whether the neuronal dynamics correlate more positively with the target movement after training compared with before training. To quantify the similarity between the neuronal dynamics and the target output, we calculate for each of the 10 training sessions (see above) the Pearson correlation coefficient of the neuronal dynamics between each neuron in the recurrent network and the target output. In Fig. S1D, we plot the mean Pearson correlation coefficient across all neurons for the case in which all gains are set to 1 (i.e., before training) and for each of the 10 learned gain patterns (i.e., after training). There is a significant (with a p-value of $p \approx 0.002$) change in the mean Pearson correlation coefficient before training versus after training using a paired Wilcoxon signed rank one-sided test. For the gain pattern that produces the largest change in the mean correlation coefficient (see the grey line in the left panel of Fig. S1D), we plot the distribution of changes in correlation coefficients for all neurons (see the right panel of Fig. S1D). We see that most values are larger than 0, so the neuronal dynamics become more positively correlated with the target output after learning. We also show an example of a substantial change in the neuronal dynamics of one neuron.

For the same task as that shown in Fig. 1D, we also use an alternative learning rule (Eqns. (10) and (11)), where learning automatically stops when the network output error becomes sufficiently low (see Section 1.8). We plot the error reduction in Fig. S1E in orange. Using this alternative learning rule, the error reaches a smaller value for this task (compare the orange curve to the red curve in Fig. S1E) and learning stops after approximately $10,000$ training iterations on average.

In another computational experiment, we train the network on the same task but instead use a ramping input to the network (simulating preparatory activity prior to movement onset [2,3]) and train the neuronal gains so that the network output generates the target. We use the same ramping input function that was used in Ref. [2], namely $\exp(t/\tau_{\text{on}})$ if $t < 0$ s and $\exp(-t/\tau_{\text{off}})$ after movement onset ($t \geq 0$), with an onset time $\tau_{\text{on}} = 400$ ms and an offset time $\tau_{\text{off}} = 2$ ms. Any gain changes resulting from learning now also affect the neuronal activity state at $t = 0$ (i.e., at movement onset). We again run 10 independent training sessions, and we observe learning results that are qualitatively similar to those above (see blue curve in Fig. S1E).

For the same task, we also train a 'chaotic' [15] variant of our network model (see Section 1.3, where we describe how we construct such a model) and apply the same training method that we described above. We use the first 0.5 s of network activity and we again change only the neuronal gains during training. We run 10 independent training sessions, and we observe a very similar error reduction over training iterations (see black curve in Fig. S1E) as we saw in Fig. 1D (compare black and red curves in Fig. S1E).

In another computational experiment, we generate 10 different target muscle activities (see Section 1.4) and independently train the neuronal gains for a network of 200 neurons, as we described earlier in this section using our learning rule Eqn. (8) (see the red curve in Fig. S1G). As a control to compare the performance of training neuronal gains, for the same 10 target movements, we independently train a rank-one perturbation of the synaptic weight matrix for each movement. Specifically, for each of the 10 movements, we learn vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^{200\times 1}$ to reduce the error between the network output, which we obtain from the neuronal dynamics in Eqn. (1) with $\boldsymbol{W}$ replaced by $\boldsymbol{W} + \boldsymbol{u}\boldsymbol{v}^\top$, and the target movement. We use Eqn. (8) to independently train the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, where g_i and $\bar{g}_i$ are replaced, respectively, by u_i and $\bar{u}_i$ and by v_i and $\bar{v}_i$. When training the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, we set all gains to 1. We find that by training with gain modulation, which is the focus of our paper, we reduce the error at a substantially faster rate compared to the training method of using a rank-one perturbation. (Compare the blue and red curves in Fig. S1G.)

In a final computational experiment, we train a network on the same task as the one that we showed in Figs. 1D–F, but with $r_0 = 5$ Hz. We plot these results in Fig. S2.

1.11 Simulation details for Fig. 2 and Fig. S3

For coarse-grained (i.e., grouped) gain modulation, we generate n random (modulatory) groups, and we independently modulate each group using one external 'modulatory unit'. Our generation mechanism proceeds 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 based on previous training, we obtain groups by applying k-means clustering to 10 gain patterns obtained from 10 independent training sessions (using neuron-specific control) on the same target.

For the same task as in Fig. 1, we plot the results of the above random and specialized groupings (as well as the neuron-specific result from Fig. 1D) in Fig. S3A. The readout weights are the same as those in Fig. 1.

We now give details for Figs. 2B,C and Figs. S3B,C. We generate 5 different target outputs and run 10 independent training sessions for each target. For the random groupings, we use different independently-generated random groupings for each simulation. However, for the specialized groupings, for a specified number of groups, we use the same grouping in all simulations. We plot the results of using 10 or 20 groups with either random or specialized groupings in Figs. 2B,C and Figs. S3B,C. When obtaining specialized groupings shared by multiple movements (i.e., we use the same grouping for learning multiple movements), as plotted in Fig. 2C and Figs. S3B,C, we use k-means clustering across all the gain patterns that we obtain using neuron-specific modulation for each of the movements. We also use the specialized grouping that we obtain for 20 groups shared across 5 movements to learn 10 hitherto-untrained movements. We plot these results in Fig. S3D.

For the same 5 targets that we just described above, we consider various different numbers of groups (determined randomly using the above procedure) 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 network generates the same initial output with all gains set to 1. The readout weights remain fixed throughout training. We plot these results in Fig. 2D and Figs. S3E–H. We use the Tukey style for the whiskers in the box plots.

We now give details for Figs. 2E,F. For multiple readout units, we generate 10 different initial network outputs and targets for each readout unit. For example, for 2 readout units, we generate 10 different initial and target outputs for each of units 1 and 2. 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 movements for all 3 networks and each number of random groups. We thus fit readout weights so that each network generates the same initial output with all gains set to 1. The readout weights remain fixed throughout training. We now use $60,000$ (instead of $18,000$) training iterations to ensure error saturation.

1.12 Simulation details for Fig. 3 and Figs. S4 and S5

To create libraries of learned movements, we train a network of 400 neurons and 40 random groups (see Section 1.11) on each of 100 different movements independently. (In other words, this generates 100 different gain patterns, with one for each movement.) In Fig. S4A, we plot the distribution of gains that we obtain after training across all 100 gain patterns. We plot all 100 outputs from these 100 learned gain patterns in Fig. S4B. We also generate 100 new gain patterns by sampling uniformly at random from the distribution in Fig. S4A and plot the output of each of these gain patterns in Fig. S4C. These outputs are much more homogeneous than the learned gain patterns in Fig. S4B, and they likely would not constitute a good basis set for movement generation.

For library sizes of $k \in \{1, 2, \ldots, 50\}$, we choose 100 samples of k movements (from the learned gain patterns and their outputs, as described above) uniformly at random without replacement for each k. We then fit the set of 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 $\sum_{j=1}^{k} c_j = 1$. That is, we consider convex combinations of the coefficients c_j. 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 movements, each of the 100 samples, and each k. See Section 1.6 for our description of how we calculate errors.

For each k and for each randomly-generated combination of library elements (see the paragraph immediately above), we order the 100 novel target movements based on the error between the output and the fit, and we select the one that is the 50th largest (i.e., close to the median error). We then extract the output and fit errors for this target and repeat this procedure for $k = 1, \ldots, 50$ and for each of the 100 randomly-generated combinations of library elements. We plot these results in Fig. 3C and Fig. S4G. In Fig. 3, we plot results for $k \in \{1, 2, \ldots, 20\}$; in Fig. S4, we plot results for $k \in \{1, 2, \ldots, 50\}$. Observe that there is only a small change in the errors between $k = 20$ and $k = 50$. In Fig. S4E, we plot the distribution of errors over the 100 samples for $k = 5$ and $k = 20$. Additionally, for each k and for each of the 100 target movements, we order the 100 combinations of library elements based on the error between the output and the fit, and we select the one that is the 50th largest. We then extract the output and fit errors for this combination and repeat this procedure for $k = 1, \ldots, 50$ and for each of the 100 target movements. We plot these results in Fig. S4H. This indicates that we obtain qualitatively similar results if we average over the 100 target movements or if we instead average over the 100 combinations of library elements.

We also calculate the Pearson correlation coefficient between the output and the fit errors for each k when taking the 50th largest error across the 100 novel target movements (see Fig. S4I) or across the 100 randomly-generated samples (see Fig. S4J).

Importantly, we also repeat these simulations for the baseline rate $r_0 = 5$ Hz in Eqn. (2). We plot the results of these simulations in Fig. S5, and we note that we obtain near identical results to those obtained for the baseline rate $r_0 = 20$ Hz.

1.13 Simulation details for Fig. 4 and Figs. S6 and S7

We now describe our simulations for learning target activity that lasts longer than 0.5 s. In each of these simulations, we use a network of 400 neurons and 40 random modulatory groups. (See Section 1.11 for our discussion of how we determine such groups.) We construct target movements with $\sigma = 312$ ms and $\ell = 75$ ms in Eqn. (5). We then construct both a 'fast' (0.5 s) and a 'slow' (2.5 s) variant of each movement. (Note that we are modelling the network output activity as a proxy for muscle-force dynamics. To actually generate the same movement so that it lasts 5 times longer, we need to also scale the amplitude of the target force dynamics by the factor $1/5^2 = 1/25$. We omit this scaling so that the task is more difficult, because the target activity without the scaling has a substantially larger amplitude throughout the duration of the movement.) Each movement variant has 500 evenly-spaced points (see 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 $2,000$ ms of the movement to ensure that both movements have the same length (see Fig. 4A; top right).

Details for Fig. 4B and Figs. S6A,C. We fit readout weights using least-squares regression so that with all gains set to 1, the network output approximates the fast variant. We then train gain patterns using our learning rule Eqn. (1.8) so that the network output generates the slow-movement variant. (The network initial condition and readout weights remain fixed.) We use $60,000$ training iterations. We run 10 independent training sessions for each of 10 different target movements. We plot one such movement in Fig. 4B, and we plot results of all simulations in Figs. S6A,C.

Details for Fig. 4C and Fig. S6B. We wish to obtain neuronal dynamics that are less sensitive to noisy network initial conditions than those that are generated from gain patterns obtained from our learning rule. For example, in Fig. 4B, the neuronal activity has decayed substantially towards 0 after approximately 0.5 s, even though the output activity is close to its maximum value. We therefore perform the task that we described in the paragraph above (i.e., generating a slow-movement variant by changing the neuronal gains) using a gradient descent-training procedure using gradients that we obtained from back-propagation [38]. 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 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 composed of the squared error between actual network outputs (fast and slow) and target outputs (fast and slow) plus the Euclidean 2-norm of the readout weights, where the latter acts as a regularizer. We run gradient descent for 500 iterations, well after the cost has stopped decreasing.

Using the target movement from Fig. 4B, we plot the output of the back-propagation training procedure in Fig. 4C, and we plot results of all simulations in Figs. S6B,D on the same 10 target movements as used in Fig. S6A. In Fig. S6G, for the outputs in Figs. 4B,C, we add white Gaussian noise with a signal-to-noise ratio of 4 dB to the network initial condition. We observe that the outputs from the back-propagation training procedure are less sensitive to noisy initial conditions than the outputs from the learning rule.

Details for Figs. S6H–J. In these simulations, we train a single gain pattern that is shared by n different movements, which each last 2.5 s and where each movement corresponds to a different network initial condition. To generate a collection of n such initial conditions, in which each initial condition evokes neuronal activity of approximately equal amplitude at the baseline condition (i.e., with all gains set to 1), we randomly rotate the top n eigenvectors of the observability Gramian of the matrix $\boldsymbol{W} - \boldsymbol{I}$ [2]. Specifically, we do this by creating a matrix of n columns—one for each these n eigenvectors—and postmultiplying this matrix by a random $n \times n$ orthogonal matrix (obtained via a QR decomposition of a random matrix with elements drawn from a normal distribution with mean 1 and standard deviation 1). We plot the results as a function of the number n of movement/initial condition pairs (see Figs. S6H,I) for 10 independent draws of the initial conditions that we just described. We use the Tukey style for the whiskers in the box plot.

Given n initial conditions, we also uniformly randomly choose n fast target movements and their slow counterparts out of a fixed set of 10 different movements. We then train a network to generate the correct fast and slow target movements by optimising a single set of readout weights and a set of n gain patterns for the slow variants (where we set the gains for each of the fast variants to 1). We train using the same gradient-descent method with back-propagation that we described earlier in this section.

Details for Fig. S7A. For each of the 10 trained movements in Figs. 6A,B, we extract the mean minimum error across all simulations for both the outputs obtained via our learning rule (see Fig. S 6A) and the outputs obtained via back-propagation (see Fig. S 6B). We then linearly interpolate between the learned gain patterns for the fast and slow outputs and calculate the error (see Section 1.6) between the output and the target movement at the interpolated speed. We calculate these errors for many interpolated movement durations between 0.5 s and 2.5 s, and we plot the mean errors for both our learning rule and the back-propagation training in Fig. S7A. We also show an example output that lasts 1.5 s.

Details for Figs. 4D,E and Figs. S 7B–D. To demonstrate that gain modulation can provide effective smooth control of movement speed for multiple network initial conditions, we train networks to generate a pair of target movements in response to a corresponding pair of orthogonal initial conditions (see the above description of Figs. S6H–J) at fast and slow speeds (as above) and also at each of 5 intermediate, evenly-spaced speeds in between these extremes. To do this, we parametrize the gain pattern of speed 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 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 $s \in \{2, \ldots, 6\}$), and a single set of readout weights. For a given speed s, we use the gain pattern $\boldsymbol{g}_s$ for both movements.

We plot the 7 learned gain patterns and their corresponding outputs for each initial condition in Figs. S 7B,D. Note that interpolating between the fast and slow gain patterns generates both movements at any intermediate speed (see Fig. S7C). We show examples of interpolating between the fast and slow gain patterns for 5 (of the 40) modulatory groups in Fig. 4D, and we plot outputs at 5 evenly-spaced speeds in Fig. 4E for both initial conditions.

Details for Fig. 4F and Figs. S7E,F. Here, we simultaneously train gain patterns for controlling different movements (i.e., different movement shapes) and their speed. We simultaneously train the network (using back-propagation, as discussed above) to generate each of 10 different movement shapes at 7 different, evenly-spaced speeds (ranging from the fast variant to the slow variant) using the same network initial condition. Importantly, 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 discussion above) the gain pattern that encodes speed s (with $s \in \{1, \ldots, 7\}$) 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 (see above), and a single set of readout weights. In Fig. S7E, we plot the gain patterns that we obtain for controlling the movement speeds at each of the 7 trained speeds. In Fig. S7F, we plot the outputs of each of the 10 gain patterns for movement shape at each of 5 interpolated speeds between the fast and the slow gain patterns. In Fig. 4F, we plot 2 example movement shapes at 3 interpolated speeds.

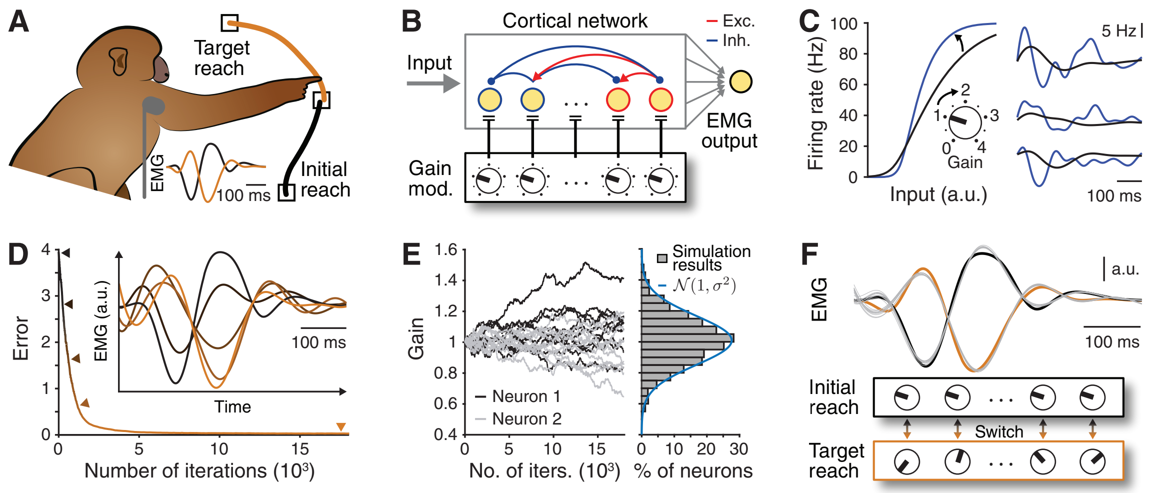

Fig. 1: **Controlling network activity through neuron-specific gain modulation**. (**A**) Example of a reaching task, with illustrative electromyograms (EMG) of muscle dynamics for two reaches (in orange and black). (**B**) Schematic of our model (see the text and Section 1.10). (**C**) Changing the slope of the input–output gain function (left) uniformly for all neurons from (black) 1 to (blue) 2 has pronounced effects on neuronal activity (right); 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. (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 for the initial and the new gain patterns for 10 noisy initial conditions (grey curves) compared to both targets (black and orange).

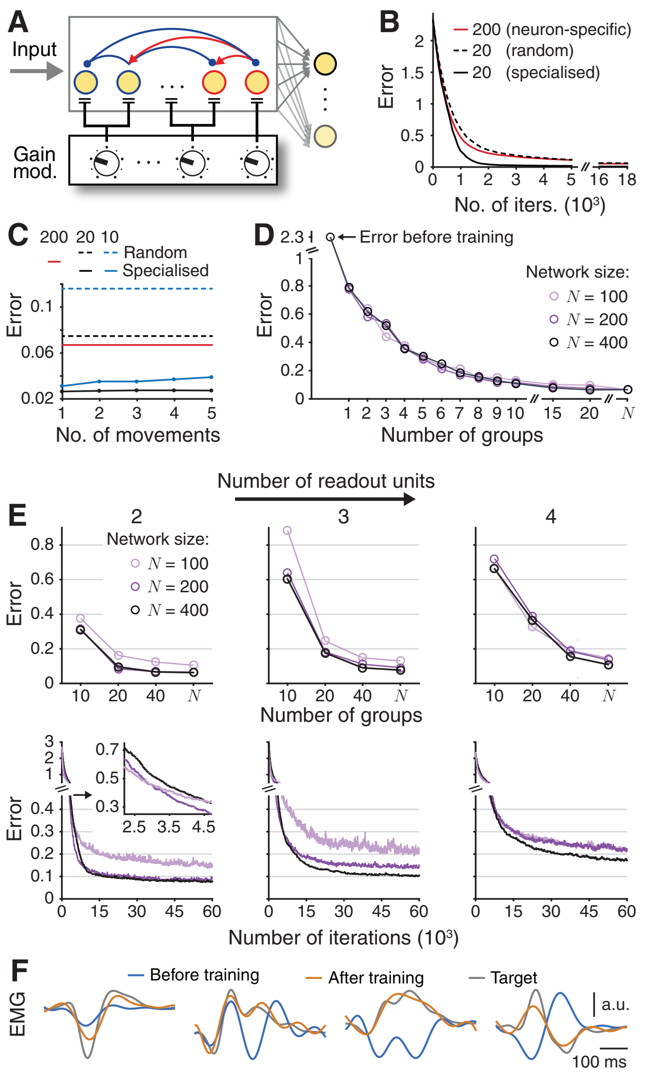

Fig. 2: **Controlling network activity through coarse, group-based gain modulation**. (**A**) We identically modulate neurons within each group (see Section 1.11), and target outputs may involve multiple readout units. (**B**) Mean error during training for 20 random, 20 specialized, and 200 (i.e., neuron-specific) groups (see Section 1.11). (**C**) Mean minimum errors after training using specialized groupings. The same groupings are used for different numbers of 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.

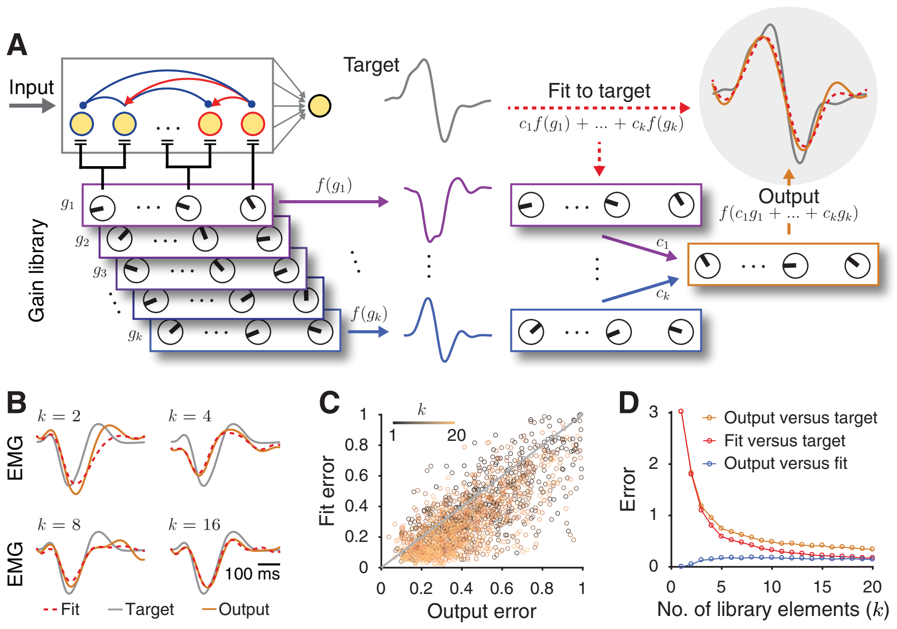

Fig. 3: **Gain patterns can provide motor primitives for novel movements**. (**A**) Schematic of a learned library of gain patterns ($g_1, \ldots, g_k$, which colour from purple to blue) and a combination $c_1 f(g_1) + \ldots + c_k f(g_k)$ of their outputs that we fit (red dashed curve) to a novel target (grey curve). (Upper right) The output (orange) $f(c_1 g_1 + \ldots + c_k g_k)$ 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 Section 1.12) (**B**) Example target, fit, and output (grey, red dashed, and orange curves, respectively) producing the median output error using $k = 2$, $k = 4$, $k = 8$, and $k = 16$ library elements. (**C**) Fit error versus the output error for 100 randomly-generated combinations (see Section 1.12) of k library elements for $k = 1, \ldots, 20$. Each point represents the median error across 100 novel target movements. We show the identity line in grey. (**D**) Median errors of the 100 randomly-generated combinations of k library elements versus the number of library elements.

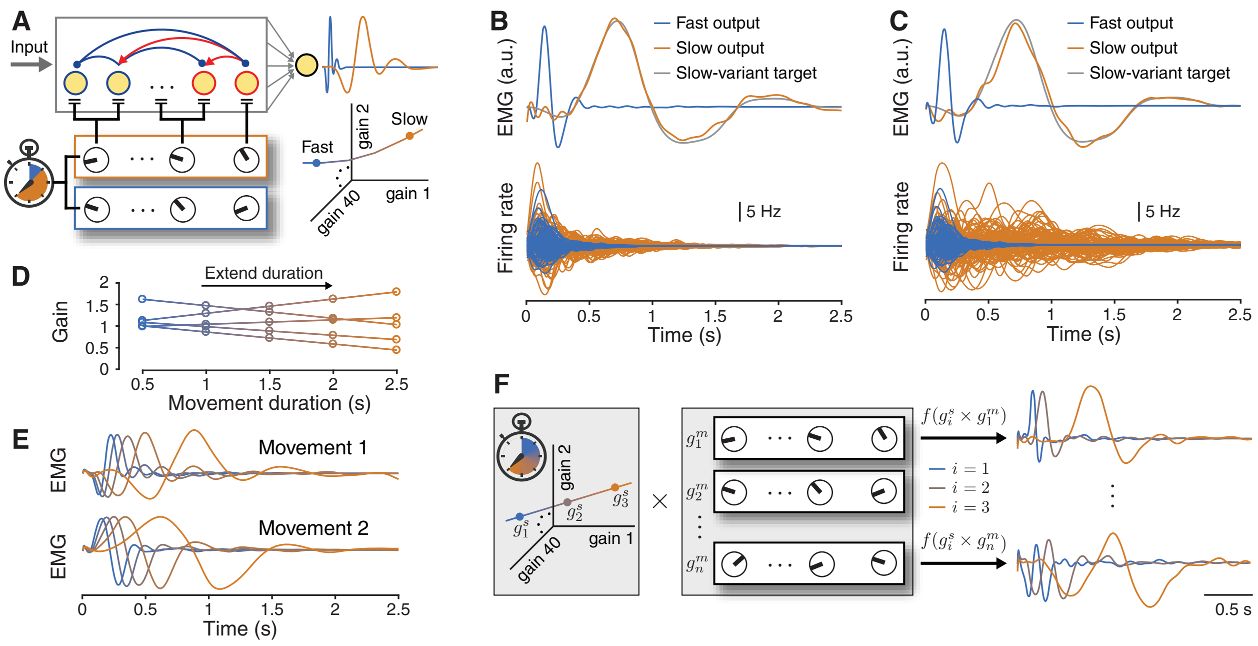

Fig. 4: **Gain modulation can control movement speed**. (**A**) Schematic of gain patterns for fast (0.5 s) and slow (2.5 s) movement variants. (Here and throughout the figure, we show the former in blue and the latter in orange.) We train a 400-neuron network using 40 random modulatory groups for all simulations. (See Section 1.13 for details.) (**B**, Top) We train a network to extend its output from a fast to a slow-movement variant using our local learning rule. (Bottom) Example dynamics of 50 excitatory and 50 inhibitory neurons for both fast and slow speed variants. (**C**) The same as panel (B) but using a back-propagation training algorithm (see Section 1.13). (**D**) A linearized gain manifold for speed control (see the main text) for 5 example modulation groups and 5 speeds trained on two initial conditions. (**E**) Both outputs for the 5 evenly-spaced speeds from panel (D). (**F**) 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 different speeds.

Supplementary Figures

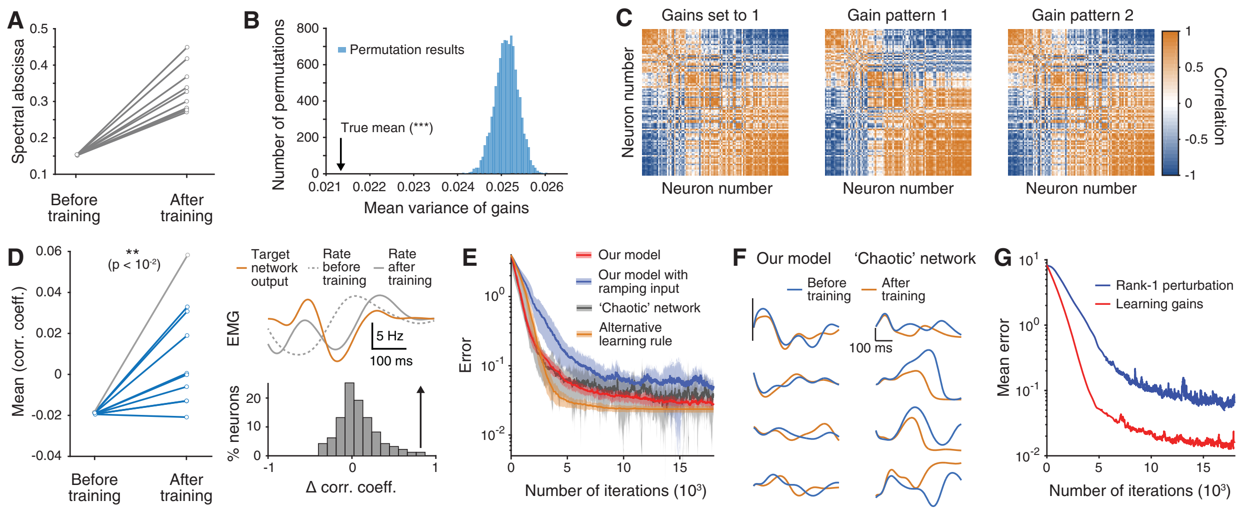

Fig. S1: **Further effects of neuron-specific gain modulation**. (**A**) Changes in the largest real part in the spectrum of $\boldsymbol{W} \times \mathrm{diag}(\boldsymbol{g})$ resulting from 10 different training sessions (see Section 1.10). Although this change appears substantial, the resulting neuronal activity does not change dramatically. (For example, see panels (C) and (F).) (**B**) The mean variance of the gains across neurons for 10 training sessions (arrow) and the distribution of mean variances with $10,000$ instances of gains shuffled uniformly at random across neurons and training sessions. (The p-value is $p < 10^{-4}$; see Section 1.10.) (**C**) Correlation matrices of the activity for all pairs of neurons with (left) all gains set to 1 and (centre and right) two independently learned gain patterns for the task in Fig. 1D. The order of neurons is the same in all three matrices. There is no substantial change in Pearson correlation between pairs of neurons as a result of training. (**D**, Left) The mean Pearson correlation coefficient between the neuronal firing rates and the target increases after training. (We show 10 training sessions.) (Bottom right) Example change in Pearson correlation coefficients between neuronal firing rates and the target after training for the trial in grey in the left panel. (Top right) Example of a substantial change in the dynamics of one neuron after training. (**E**) Mean error during training for our model (red) (see Fig. 1D), our model with a biologically realistic ramping input (blue), a 'chaotic' recurrent network model (grey), and our model when using the alternative learning rule from Eqn. (10) (orange) (see Section 1.10). Shading indicates one standard deviation. (**F**) The firing rates of 5 example neurons before and after training in (left) our model and (right) the 'chaotic' network. The black vertical bars on the left and right indicate 5 Hz and 10 Hz, respectively. (**G**) Mean error during training when independently learning 10 different target movements using our learning rule when training the neuronal gains (red) or training a rank-one perturbation of the synaptic weight matrix (blue) (see Section 1.10).

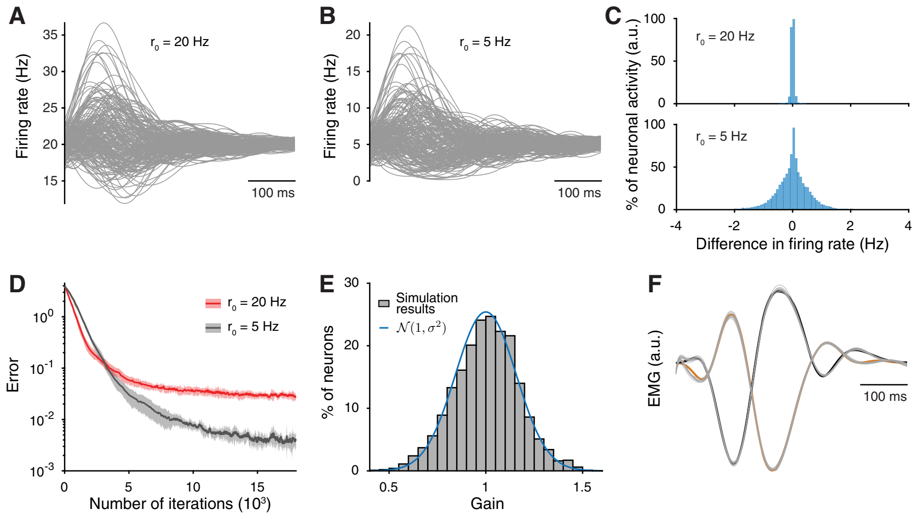

Fig. S2: **Neuron-specific gain modulation with $r_0 = 5$ Hz.** (**A**) Firing rate of all neurons in a 200-neuron network with $r_0 = 20$ Hz. (**B**) Firing rate of all neurons in the same 200-neuron network with $r_0 = 5$ Hz. (**C**) A histogram of the difference in firing rates across all neurons compared to the case of a network with a linear gain function (i.e., $f(x_i; g_i) = g_i x_i$ in Eqn. (1)) with (top) $r_0 = 20$ Hz and (bottom) $r_0 = 5$ Hz. For $r_0 = 20$ Hz, the neuronal dynamics are much more similar to the neuronal dynamics from a linear gain function than is the case for $r_0 = 5$ Hz. (**D**) Mean error during training for our model with $r_0 = 20$ Hz (red) (see Fig. 1D) and our model with $r_0 = 5$ Hz (grey) for the task in Fig. 1D (see Section 1.10). Shading indicates one standard deviation. With $r_0 = 5$ Hz, the network outputs achieve a lower final error after training. (**E**) Histogram of gain values after training. The blue curve is a Gaussian distribution with a mean of 1 and standard deviation of $\sigma \approx 0.157$ (i.e., the distribution that we obtained with $r_0 = 20$ Hz in Fig. 1E). The distribution of learned gains is almost identical to what we obtained with $r_0 = 20$ Hz. (**F**) Network outputs for the initial and the new gain patterns with $r_0 = 5$ Hz for 10 noisy initial conditions (grey curves) compared to the two targets (black and orange).

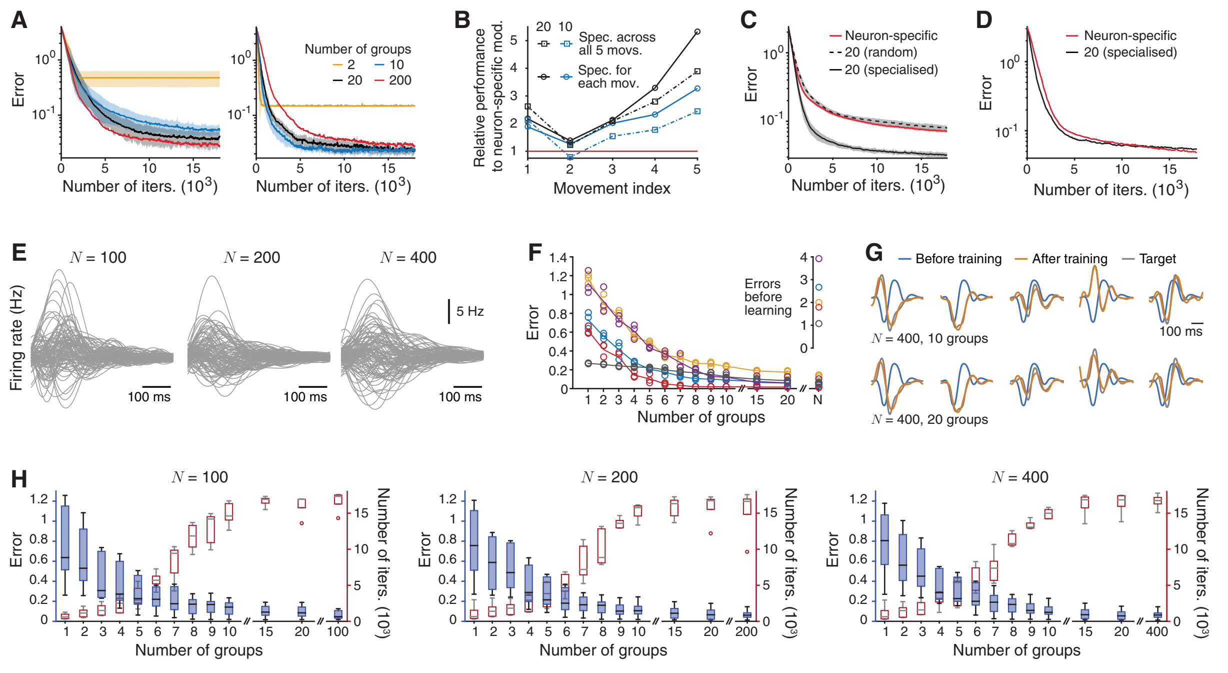

Fig. S3: **Additional results for grouped gain modulation**. (**A**) Mean error over 10 training sessions (where shading indicates one standard deviation) using (left) random and (right) specialized groupings for 2, 10, 20, and 200 (i.e., neuron-specific) groups (see Section 1.11). The target output is the same as in Fig. 1. (**B**) Relative improvement in performance compared with neuron-specific modulation for each of 5 movements when using specialized groups shared across all (squares) or for each (circles) of the 5 movements using either 10 (blue) or 20 (black) groups. A value of 2 implies that the error is 2 times smaller after training compared to neuron-specific modulation. (**C**) Mean error over 10 training sessions (where shading indicates one standard deviation) when learning 5 movements using the same set of 20 specialized groups (shared across all 5 movements), 20 random groups, and neuron-specific modulation. (**D**) Mean error over 10 training sessions when learning 10 novel movements using the specialized grouping (with 20 groups) shared across the 5 previously trained movements from panel (C). (**E**) The dynamics of 50 inhibitory and 50 excitatory neurons for each of the three different networks sizes. (**F**) The curves give the mean error over 10 training sessions and across the 3 networks for each of 5 targets. The circles represent the mean error for each network, and the different colours indicate each of 5 different target outputs (see Section 1.11). (**G**) Outputs for all five targets from the trial that produces the median error for the 400-neuron network for the cases of 10 and 20 groups. (**H**) Box plots (in blue) of the minimum error after training for different numbers of groups and the 3 different network sizes. (These are the same data that we plotted in panel (F).) We also include box plots (in red) for the minimum number of iterations required before the error is within 1 % of the minimum error.

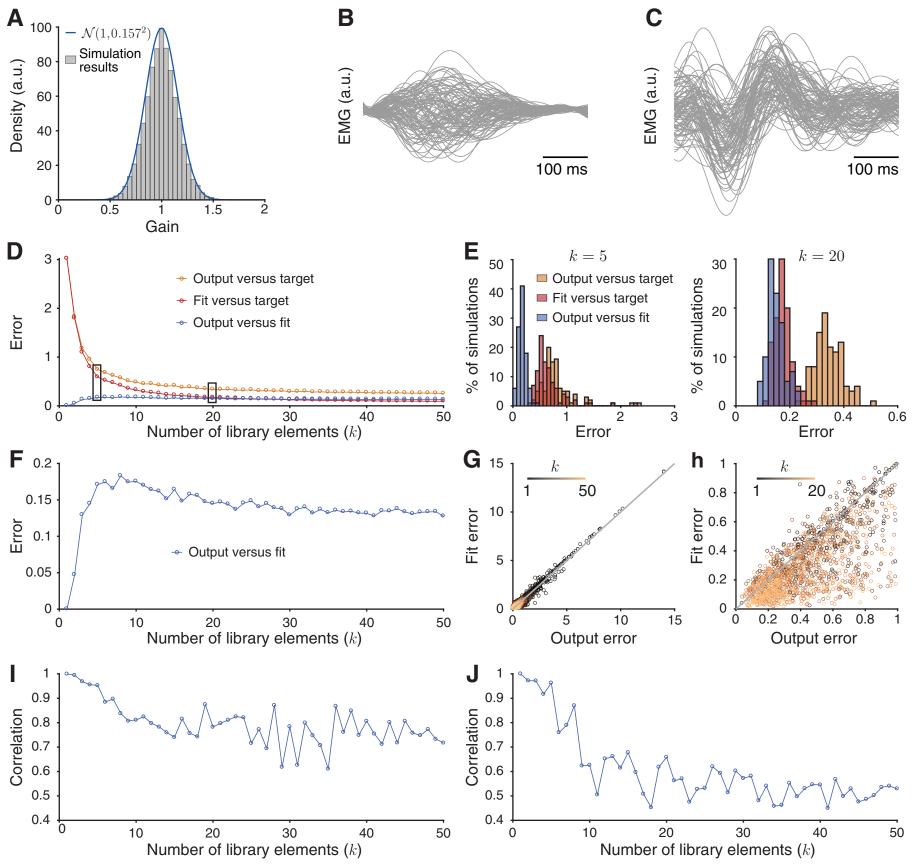

Fig. S4: **Additional results for gain patterns providing motor primitives**. (**A**) The resulting distribution of gains from training independently on each of 100 targets (see Section 1.12). The distribution of the gain patterns resembles a normal distribution (blue curve) with the same mean and variance as those found in Fig. 1E . (**B**) Each output from the 100 trained gain patterns. (**C**) Outputs of 100 randomly-generated gain patterns from the distribution in panel (A). (See Section 1.12 for details.) The outputs are substantially more homogeneous than those in panel (B) and likely would not constitute a good library for movement generation. (**D**) The same plot as in Fig. 3D but for up to $k = 50$ library elements. (**E**) The distributions of errors across 100 different libraries for (left) $k = 5$ and (right) $k = 20$. (Note the difference in horizontal-axis scales in the two plots.) (**F**) The error between the fit and the output from panel (D). (**G**) The same plot as in Fig. 3C but for $k = 1, \ldots, 50$ and with extended axes. Each point represents the median error across 100 novel target movements for each of 100 randomly-generated combinations of k library elements. We show the identity line in grey. (**H**) The same as in panel (G), but each point represents the median error across the 100 libraries for each of the 100 novel target movements. We plot these data in the square $[0, 1] \times [0, 1]$ and for $k = 1, \ldots, 20$. (**I**) For the data in panel (G), we plot the Pearson correlation coefficient between the output and the fit errors for each number of library elements. (**J**) For the data in panel (H), we plot the Pearson correlation coefficient between the output and the fit errors for each number of library elements (up to $k = 50$).

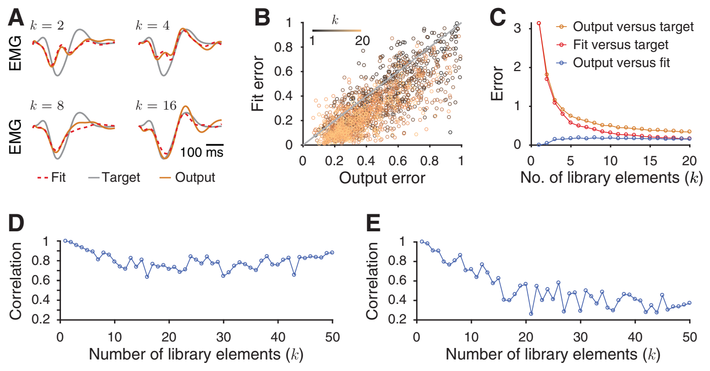

Fig. S5: **Gain patterns as motor primitives with $r_0 = 5$ Hz.** (**A**) Example target (grey), fit (dashed red), and output (orange) producing the median output error using $k = 2$, $k = 4$, $k = 8$, and $k = 16$ library elements. (**B**) Fit error versus the output error for 100 randomly-generated combinations (see Section 1.12 for a description of the generation process) of k library elements for $k = 1, \ldots, 20$. Each point represents the median error across 100 novel target movements. We show the identity line in grey. (**C**) Median errors of the 100 randomly-generated combinations of k library elements versus the number of library elements. Compare panels (A–C) of this figure with panels (B–D) in Fig. 3. (**D**) For the data in panel (B), we plot the Pearson correlation coefficient between the output and the fit errors for each number of library elements (up to $k = 50$). (**E**) The same as panel (D) but for data corresponding to the median errors for each novel target movement, rather than for each randomly-generated combination of library elements (up to $k = 50$) (see Section 1.12). Compare panels (D) and (E) of this figure with panels (I) and (J) in Fig. S4.

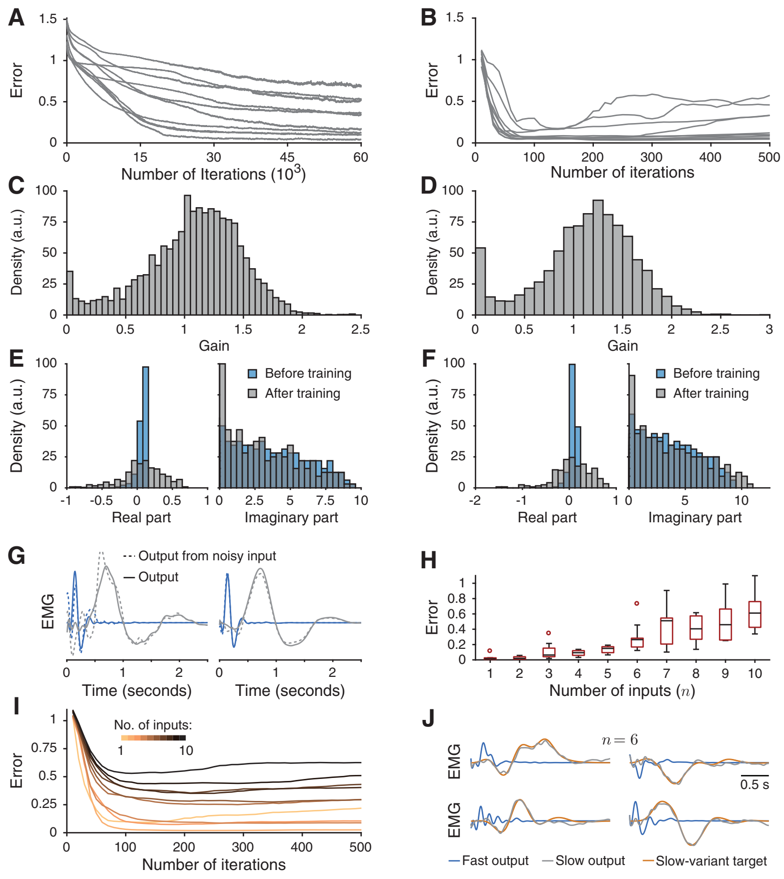

Fig. S6: **Additional results for controlling movement speeds through gain modulation**. (**A**) Mean error over 10 training sessions for 10 different movements when learning gain patterns for slow-movement variants using our reward-based learning rule (see Section 1.13). (**B**) Mean error over 10 training sessions for the same 10 movements when instead learning gain patterns for slow-movement variants using a back-propagation algorithm (see Section 1.13). (**C**) Distribution of gains for the slow-movement variants across all training sessions using our reward-based learning rule. (**D**) Distribution of gains for the slow-movement variants across all training sessions using the back-propagation algorithm. (**E**) Histograms of the real and imaginary parts of the eigenvalues of the linearized system Eqn. (1) before and after training using our reward-based rule for the example shown in Fig. 4A. (**F**) Histograms of the real and imaginary parts of the eigenvalues of the linearized system Eqn. (1) before and after training using the back-propagation algorithm for the example in Fig. 4B. (**G**) The same outputs plotted in Figs. 4A,B with white Gaussian noise, with a signal-to-noise ratio of 4dB, added to the network initial condition (see Section 1.13). (**H**) Box plot of the slow-movement-variant errors across 10 training sessions for different numbers of initial conditions. (**I**) Mean error over 10 training sessions for $n = 1, \ldots, 10$ initial conditions. (**J**) For the case of 6 initial conditions in panel (H), we plot the 4 example outputs that produce the median error for the 10 training sessions. (For each simulation, we train a 400-neuron network using 40 random modulatory groups (see Section 1.13).)

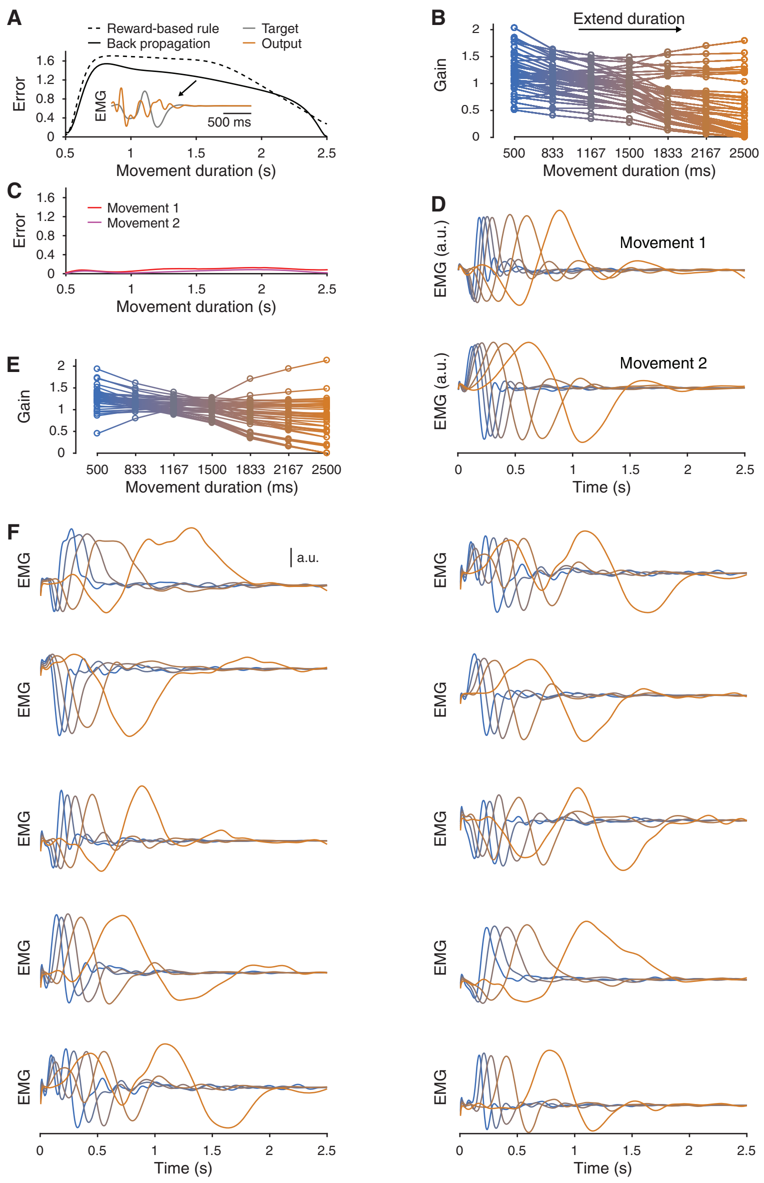

Fig. S7: **Additional results for smooth interpolation of movement speeds through gain modulation**. (**A**) Interpolation between fast and slow gain patterns does not reliably produce outputs of intermediate speeds when trained only at the fast and slow speeds (see Section 1.13). (**B**) We show the 7 optimized gain patterns for all 40 modulatory groups when training at 7 evenly-spaced speeds (see Section 1.13). (**C**) Linear interpolation between the fast and slow gain patterns successfully approximates the target output when trained at 5 intermediate speeds for 2 initial conditions. (Note that we plot these results on the same axes as in panel (A).) (**D**) Outputs for both initial conditions from the 7 trained gain patterns from panel (B). (**E**) The 7 optimized gain patterns for movement speed when jointly training gain patterns for the speed and shape of 10 movements (see Section 1.13). (**F**) Outputs at 5 interpolated speeds for all 10 movements. (For each simulation, we train a 400-neuron network using 40 random modulatory groups (see Section 1.13).)