A rapid and efficient learning rule for biological neural circuits

Eren Sezener[*1], Agnieszka Grabska-Barwińska[*1], Dimitar Kostadinov[*2], Maxime Beau[2], Sanjukta Krishnagopal[3], David Budden[1], Marcus Hutter[1], Joel Veness[1], Matthew Botvinick[1,3], Claudia Clopath[†4], Michael Häusser[†2], Peter E. Latham[†3]

[*†]Equal contribution
[1]DeepMind
[2]Wolfson Institute for Biomedical Research, University College London
[3]Gatsby Computational Neuroscience Unit, University College London
[4]Department of Bioengineering, Imperial College London

March 10, 2021

Abstract

The dominant view in neuroscience is that changes in synaptic weights underlie learning. It is unclear, however, how the brain is able to determine which synapses should change, and by how much. This uncertainty stands in sharp contrast to deep learning, where changes in weights are explicitly engineered to optimize performance. However, the main tool for doing that, backpropagation, is not biologically plausible, and networks trained with this rule tend to forget old tasks when learning new ones. Here we introduce the Dendritic Gated Network (DGN), a variant of the Gated Linear Network [1, 2], which offers a biologically plausible alternative to backpropagation. DGNs combine dendritic "gating" (whereby interneurons target dendrites to shape neuronal response) with local learning rules to yield provably efficient performance. They are significantly more data efficient than conventional artificial networks and are highly resistant to forgetting, and we show that they perform well on a variety of tasks, in some cases better than backpropagation. The DGN bears similarities to the cerebellum, where there is evidence for shaping of Purkinje cell responses by interneurons. It also makes several experimental predictions, one of which we validate with *in vivo* cerebellar imaging of mice performing a motor task.

1 Introduction

A hallmark of intelligent systems is their ability to learn. Humans, for instance, are capable of amazing feats – language acquisition and abstract reasoning being the most notable – and even fruit flies can learn simple reward associations [3, 4]. It is widely believed that learning is implemented via synaptic plasticity. But which synapses should change in response to, say the appearance of a reward, and by how much? This is especially hard to answer in humans, who have about 10^{14} synapses, but it is hard even in fruit flies, which have about 10^8 – corresponding to 100 million adjustable parameters.

One answer to this question is known: introduce a loss function (a function that measures some aspect of performance, with higher performance corresponding to lower loss), compute the gradient of the loss with respect to the weights (find the direction in weight space that yields the largest improvement in performance), and change the weights in that direction. If the weight changes are not too large, this will, on average, reduce the loss, and so improve overall performance.

This approach has been amazingly successful in artificial neural networks, and has in fact driven the deep learning revolution [5]. However, the algorithm for computing the gradient in deep networks is not directly applicable to biological systems as first pointed out by [6, 7] (see also recent reviews, [8–10]). First, to implement backpropagation [11–13], referred to simply as backprop, neurons would need to know their outgoing weights. Second, backprop requires two stages: a forward pass (for computation) and a backward pass (for learning). Moreover, in the backward pass an error signal must propagate from higher to lower areas, layer by layer (Fig. 1A), and information from the forward pass must remain in the neurons. However, biological neurons do not know their outgoing weights, and there is no evidence for a complicated, time-separated backward pass.

Backprop also leads to another problem, at least in standard deep learning setups: it adapts to the data it has seen most recently, so when learning a new task it forgets old ones [14]. This is known as catastrophic forgetting, and prevents networks trained with backprop to display the lifelong learning that comes so easily to essentially all organisms [15, 16].

Driven in part by the biological implausibility of backprop, there have been several proposals for architectures and learning rules that might be relevant to the brain. These include feedback alignment [17,18], creative use of dendrites [19,20], multiplexing [21], and methods in which the error signal is fed directly to each layer rather than propagating backwards from the output layer [22–29]. A particularly promising method that falls into the latter category is embodied in Gated Linear Networks [1, 2]. These networks, which were motivated from a machine learning rather than a neuroscience perspective, have obtained state-of-the-art results in regression and denoising [30], contextual bandit optimization [31], and transfer learning [32].

In Gated Linear Networks, the goal of every neuron, irrespective of its layer, is to predict the target output based on the input from the layer directly below it. This is very different from backprop, in which neurons in intermediate layers extract features that make it easier for subsequent layers to predict the target (compare Figs. 1A and B).

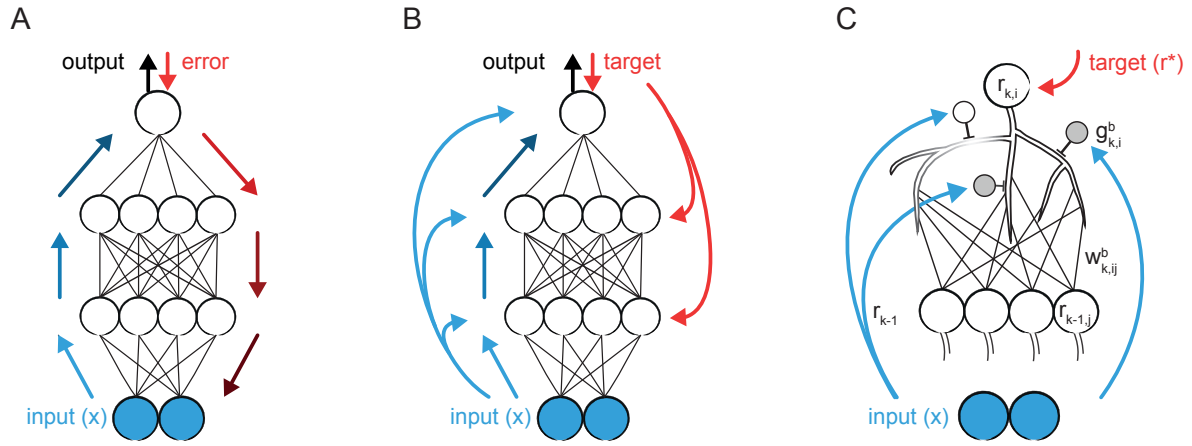

Figure 1: Comparison of multi-layer perceptrons (MLPs) and Dendrtic Gated Networks (DGNs). In all panels the blue filled circles at the bottom correspond to the input. **A**. MLP. Blue arrows show feedforward computations; red arrows show the error propagating back down. **B**. DGN. As with MLPs, information propagates up, as again shown by the blue arrows. However, rather than the error propagating down, each layer receives the target output, which it uses for learning. **C**. A single postsynaptic neuron in layer k of a DGN, along with several presynaptic neurons in layer $k-1$. Each branch gets input from all the presynaptic neurons (although this is not necessary), and those branches are gated on and off by inhibitory interneurons which receive external input. The white interneuron is active, so its corresponding branch is gated off, as indicated by the light gray branches; the gray neurons are not active, so their branches are gated on.

Gated Linear Networks are thus particularly suitable for biologically plausible learning: every neuron is essentially part of a shallow network, with no hidden layers, for which the delta rule [33] – a rule that depends only on presynaptic and postsynaptic activity, the latter relative to the target activity – is sufficient to learn.

To implement these local learning rules, the target activity is sent to every neuron, in every layer of the network (Fig. 1B, red arrows). This is typical of a large class of learning rules [22, 23, 25–29]. Completely atypical, though, is the role of the external input. It's used for gating the weights: each neuron has a bank of weights at its disposal, and the external input determines which one is used. For example, a neuron might use one set of weights when the visual input contains motion cues predominantly to the right; another set of weights when it contains motion cues predominantly to the left; and yet another when there are no motion cues at all.

Endowing each neuron with a library of weights is, of course, highly inconsistent with what we see in the brain. So instead we gate dendritic branches on and off, using inhibitory neurons, in an input-dependent manner (Fig. 1C). We thus refer to these networks as Dendritic Gated Networks (DGNs). Dendritic gating allows DGNs to represent essentially arbitrary nonlinear functions. Moreover, gating makes DGNs especially resistant to forgetting. In particular, when data comes in sufficiently separate "tasks", they can learn new ones without forgetting the old. Finally, the loss is a convex

function of the weights for each unit (see Supplementary Methods), as it is in Gated Linear Networks [1]. Convexity is an extremely useful feature, as it enables DGNs, like the Gated Linear Networks on which they are based, to learn (provably) efficiently.

Below we describe multi-layer Dendritic Gated Networks in detail – both the architecture and the learning rule. We then train them on three tasks: one on which networks trained with backprop typically exhibit catastrophic forgetting, and two relevant to the cerebellum. Finally, we show experimentally that in the cerebellum gating remains relatively stable over time – a key prediction of our model. We map the proposed learning rule and the associated architecture to cerebellum because 1) the climbing fibers provide a feedback; 2) its input-output function is relatively linear [34–36]; and 3) molecular layer interneurons could act as a gate [37–46].

2 Results

2.1 Dendritic Gated Networks

Dendritic Gated Networks, like conventional deep networks, are made up of multiple layers, with the input to each layer consisting of a linear combination of the activity in the previous layer. Unlike conventional deep networks, though, the weights are controlled by external input, via gating functions, denoted $g(\mathbf{x})$ (Fig. 1B); those functions are implemented via dendritic branches (Fig. 1C).

This results in the following network equations. The activity (i.e., the instantaneous firing rate) of the ith neuron in layer k, denoted $r_{k,i}$, is

$$r_{k,i} = \phi\left(\sum_{b=1}^{B_{k,i}} g_{k,i}^{b}(\mathbf{x}) \sum_{j=0}^{n_{k-1}} w_{k,ij}^{b} h_{k-1,j}\right), \tag{1a}$$

with the synaptic drive, $h_{k-1,j}$, given in terms of $r_{k-1,j}$ as

$$h_{k-1,j} = \phi^{-1}\big(r_{k-1,j}\big). \tag{1b}$$

Here $\phi(\cdot)$ is the activation function (either identity or sigmoid), $\mathbf{x}$ is the input of the network ($\mathbf{x}$ is an n-dimensional vector, $\mathbf{x} = (x_1, x_2, ..., x_n)$), $r_{k,i}$ is the activity of ith neuron on layer k (with $r_{k,0}$ set to 1 to allow for a bias term), $B_{k,i}$ is the number of branches of neuron i in layer k, $w_{k,ij}^{b}$ is the weight from neuron j in layer $k-1$ to the bth branch of neuron i in layer k, n_k is the number of neurons in layer k, and $g_{k,i}^{b}(\mathbf{x})$ is the binary gating variable; it is either 1 (in which case the bth branch of the ith neuron is gated on) or 0 (in which case it is gated off). There are K layers, so k runs from 1 to K. The input to the bottom layer is $\mathbf{x}$. The mapping from the input, $\mathbf{x}$, to the gating variable, $g_{k,i}^{b}(\mathbf{x})$, is not learned; instead, it is pre-specified, and does not change with time. In all of our simulations we use random half-space gating [1]; that is,

$$g_{k,i}^{b}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{v}_{k,i}^{b} \cdot \mathbf{x} \geq \theta_{k,i}^{b} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{v}^b_{k,i}$ and $\theta^b_{k,i}$ are sampled randomly and kept fixed throughout learning (see Methods), and "$\cdot$" is the standard dot product.

In Dendritic Gated Networks, the goal of each neuron is to predict the target, denoted r^*. To do that, the weights, $w^b_{k,ij}$, are modified to reduce the loss, $\ell_k(r^*, r_{k,i})$. For weight updates we use gradient descent,

$$\Delta w^b_{k,ij} = -\eta \, \frac{\partial \ell(r^*, r_{k,i})}{\partial w^b_{k,ij}} \tag{3}$$

where $\eta > 0$ is the learning rate, and the updates are performed after each sample. The precise form of the loss can influence both the speed of learning and the asymptotic performance, but conceptually we should just think of it as some distance between r^* and $r_{k,i}$. In most of the simulations, we assume ϕ is the identity ($r_{k,i} = h_{k,i}$) and we use quadratic loss

$$\ell(r^*, r_{k,i}) = \frac{1}{2}(r^* - r_{k,i})^2 \,, \tag{4}$$

so the update rule becomes

$$\Delta w^b_{k,ij} = \eta \, g^b_{k,i}(\mathbf{x})(r^* - r_{k,i}) \, h_{k-1,j} \,. \tag{5}$$

This has the form of a gated version of the delta rule [33]. See Methods, Sec. 4.1 for the alternative formulation, which we use for classification problems.

2.2 Simulations

Equations (1) and (2) for the network dynamics and Eq. (3) for learning constitute a complete description of our model. For a given problem, we just need to choose a target input-output relationship (a mapping from $\mathbf{x}$ to r^*) and specify the loss functions, $\ell(r^*, r_{k,i})$. Here we consider two tasks. The first two (catastrophic forgetting) tasks are classification, for which we use a sigmoid activation for ϕ and log loss (Methods, Sec. 4.1); the remaining (cerebellar) tasks are regression, where we use an identity activation and quadratic loss (Sec. 2.1).

DGNs can mitigate catastrophic forgetting.

Animals are able to acquire new skills throughout life, seemingly without compromising their ability to solve previously learned tasks [15, 16]. Standard networks do not share this ability: when trained on two tasks in a row, they tend to forget the first one (see Fig. 2, second row). This phenomenon, known as "catastrophic forgetting", is an old problem [47–49], and many algorithms have been developed to address it. These algorithms typically fall into two categories. The first involves replaying previously seen tasks during training [49–51]. The second involves explicitly maintaining additional sets of model parameters related to previously learned tasks. Examples include freezing a subset of weights [52, 53], dynamically adjusting learning rates [54], and augmenting the loss with regularization terms with respect to past parameters [55–57]. A limitation of these approaches (aside from additional algorithmic and computational complexity) is that they require task boundaries to be provided or accurately inferred.

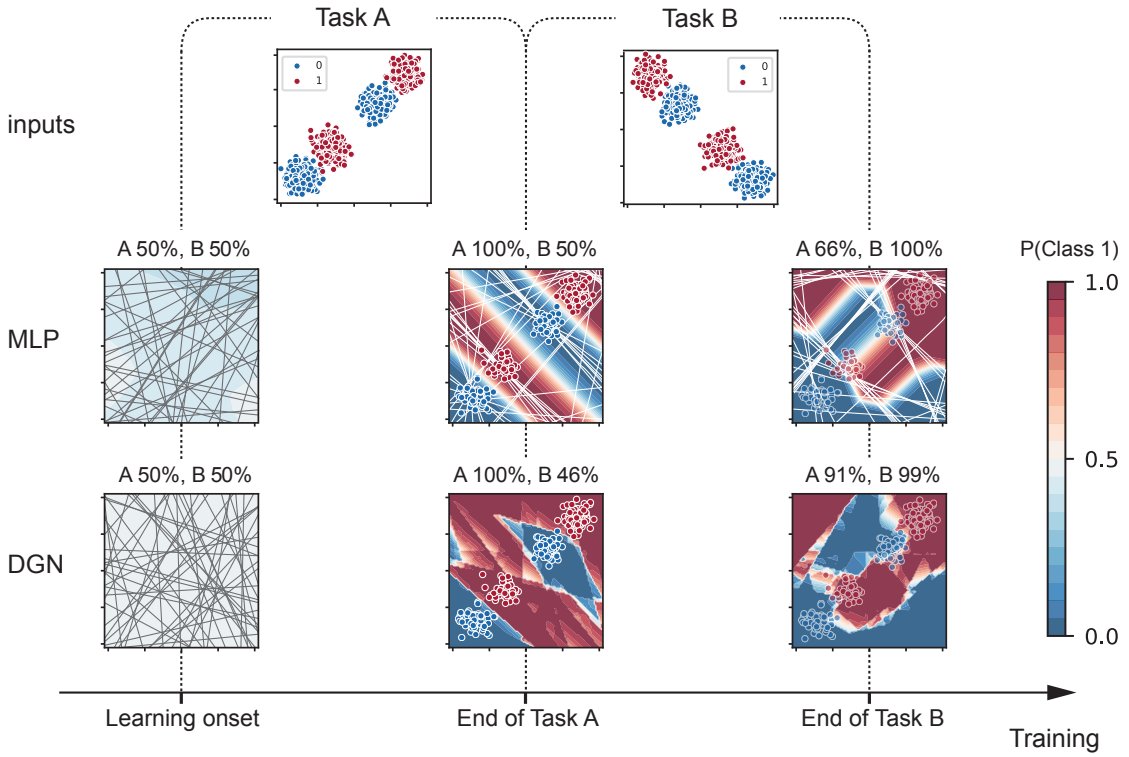

Figure 2: Comparison of the DGN to a standard multi-layer perceptron (MLP) trained with backprop. Each point on the square has to be classified as "red" ("Class 1") or "blue" ("Class 0"). We consider a scenario common in the real world (but difficult for standard networks): the data comes in two separate tasks, as shown in the first row. We trained a 2-layer MLP (second row) and a 2-layer DGN (third row) on the two tasks. The output of the network is the probability of each class, as indicated by color; the percentages report the accuracy for each of the tasks. The MLP uses ReLU activation functions, so each neuron has an effective gating; the boundaries of those gates are shown in gray. The boundaries move with learning, and are plotted at the end of training of each of the tasks (white lines). The boundaries of the DGN do not move, so we plot them only in the first column. After training on Task A, most of the boundaries in the MLP are aligned at -45 degrees, parallel to the decision boundaries, which allows the network to perfectly separate the two classes. In the DGN, the boundaries do not change, but the network also perfectly separates the two classes. However, after training on Task B, the DGN retains high performance on Task A (91%), while the MLP's performance drops to 66%. That's because many of the boundaries changed to the orthogonal direction (45 degrees). For the DGN, on the other hand, changes to the network were much more local, allowing it to retain the memory of the old task (see samples from Task A overlaid on all panels) while accommodating the new one. The MLP has 50 neurons in the hidden layer; the DGN has 5 neurons on 10 dendritic branches in the hidden layer.

Unlike contemporary neural networks, the DGN architecture and learning rule is naturally robust to catastrophic forgetting without any modifications or knowledge of task boundaries. In Fig. 2 we illustrate the mechanism behind this robustness, and show how it differs from a standard multi-layer perceptron on a single example. To demonstrate this in a more challenging task, we train a DGN on the pixel-permuted MNIST continual learning benchmark [55, 58]. In this benchmark, the network has to learn random permutations of the input pixels, with the random permutation changing every 60,000 trials (see Methods for additional details). We compare the DGN to a multi layer perceptron (MLP) with and without elastic weight consolidation (EWC) [55] as per the original papers [2, 55]. EWC is a highly-effective method explicitly designed to prevent catastrophic forgetting by storing parameters of previously seen tasks. However, it has a much more complicated architecture, and it must be supplied with task boundaries, so it receives more information than the DGN.

Because MNIST has 10 digits, we train 10 different DGNs. The alternative would be a single DGN where each unit has a 10 dimensional output corresponding to the class probabilities. However, this setting is not as biologically plausible, so we did not use it. Each of the 10 networks contains 3 layers, with 100, 20, and 1 neurons per layer, and has 10 dendritic branches per neuron. The targets are categorical (1 if the digit is present, 0 if it is not), so we use Bernoulli log loss rather than quadratic loss (see Methods, Sec. 4.1). We use 1000, 200, and 10 neurons per layer for the MLP (so that the number of neurons match the number used for the DGN), with cross entropy loss, both with and without elastic weight consolidation, and optimize the learning rates separately for each network.

Figure 3 shows the learning and retention performance of the DGN, with the MLP and EWC networks included primarily as benchmarks (recall that neither is biologically plausible). In Fig. 3A we plot performance on each task for the three networks; as can be seen, performance is virtually identical. In Fig. 3B we investigate resistance to forgetting, by plotting the performance on the first task as the nine subsequent tasks are learned. The EWC network retains its original performance almost perfectly, the MLP forgets rapidly, and the DGN is in-between. It is not surprising that the EWC does well, as it was tailored to this task, and in particular it was explicitly given task boundaries. Somewhat more surprising is the performance of the DGN, which had none of these advantages but still forgets much more slowly than the MLP. The DGN also learns new tasks more rapidly than either the EWC or MLP networks (Supplementary Figure S1), possibly because of its convex loss function.

Mapping DGNs to the Cerebellum

For the next two simulations we consider computations that can be mapped onto cerebellar circuitry. We focus on the cerebellum for several reasons: it is highly experimentally accessible; its architecture is well characterized; there is a clear feedback signal (the climbing fiber) to the Purkinje cells (the cerebellar neurons principally involved in learning); its input-output function is relatively linear [34–36]; and molecular layer interneurons play a major role in shaping Purkinje cell responses [37–43, 45], and can influence climbing fiber-mediated dendritic calcium signals in Purkinje cells [44, 46].

Both classic and more modern theoretical studies in the cerebellum have focused

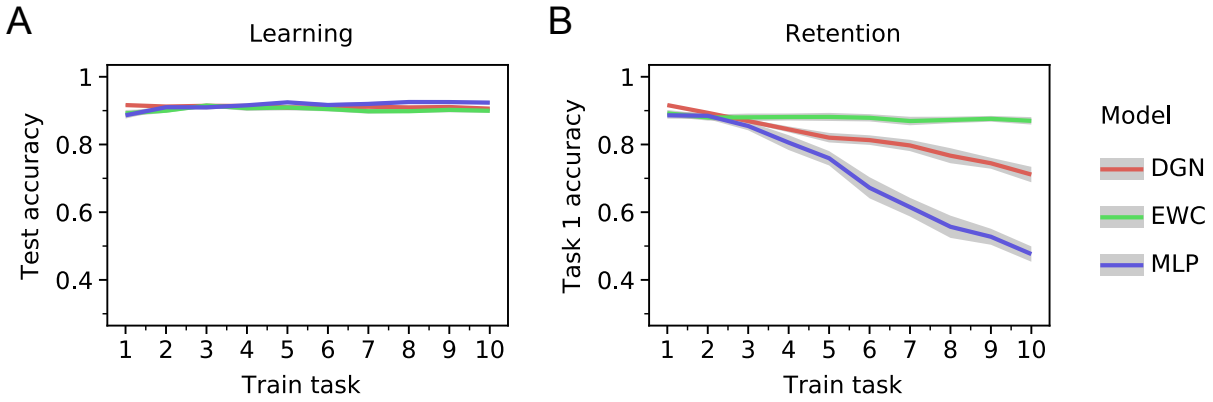

Figure 3: Learning and retention on the permuted MNIST task. The tasks are learned sequentially in a continual learning setup. **A**. Performance (on test data) for each of the 10 tasks, where a "task" corresponds to a random permutation of the pixels. **B**. Performance on the first task after each of nine new tasks is learned. As discussed, the MLP is especially bad at this task. The EWC is much better, to a large extent because it was provided with extra information: the task boundaries. Even though the DGN was not given that information, it forgets a factor of two more slowly than the MLP. Error bars in both plots denote 95% confidence over 20 random seeds.

on the the cerebellar cortex, modelling it as a one-layer feedforward network [59–63]. In this view, the parallel fibers project to Purkinje cells, and their synaptic weights are adjusted under the feedback signal from the climbing fibers. This picture, however, is an over-simplification, as Purkinje cells do not directly influence downstream structures. Instead, they project to the cerebellar nucleus neurons, which constitute the ultimate output of the cerebellum (see Fig. 4). The fact that Purkinje cells form a hidden layer, combined with the observed plasticity in the Purkinje cell to cerebellar nucleus synapses [64–68], means most learning rules tailored to one-layer networks, including the delta rule, cannot be used to train the network.

We propose instead that the cerebellum acts as a two layer DGN comprised of Purkinje cells as the first, hidden layer and the cerebellar nucleus as the second, output layer (Fig. 4). Parallel fibers provide the input to both the input layer (Purkinje cells) as well as the gates, represented by molecular layer interneurons, that control learning in individual Purkinje cell dendrites. For the second layer of the DGN, we use a non-gated linear neuron rather than a gated neuron. This is because the unique biophysical features of cerebellar nuclear neurons allow them to integrate input linearly [69]. Note that we can keep the DGN formulation given in Eq. (1); in the second layer we just use one branch ($B_{2,i} = 1$) which is always gated on ($g^1_{2,i}(\mathbf{x}) = 1$). Finally, the climbing fibers provide the feedback signal to Purkinje cells and cerebellar nuclear neurons. In our formulation, climbing fiber feedback signals the target, allowing each neuron to compute its own local error by comparing the target to its output ($r_{k,i}$). This formulation is a departure from the strict error-coding role that is traditionally attributed to climbing fibers, but is consistent with a growing body of evidence that climbing fibers signal a variety of sensorimotor and cognitive predictions [70].

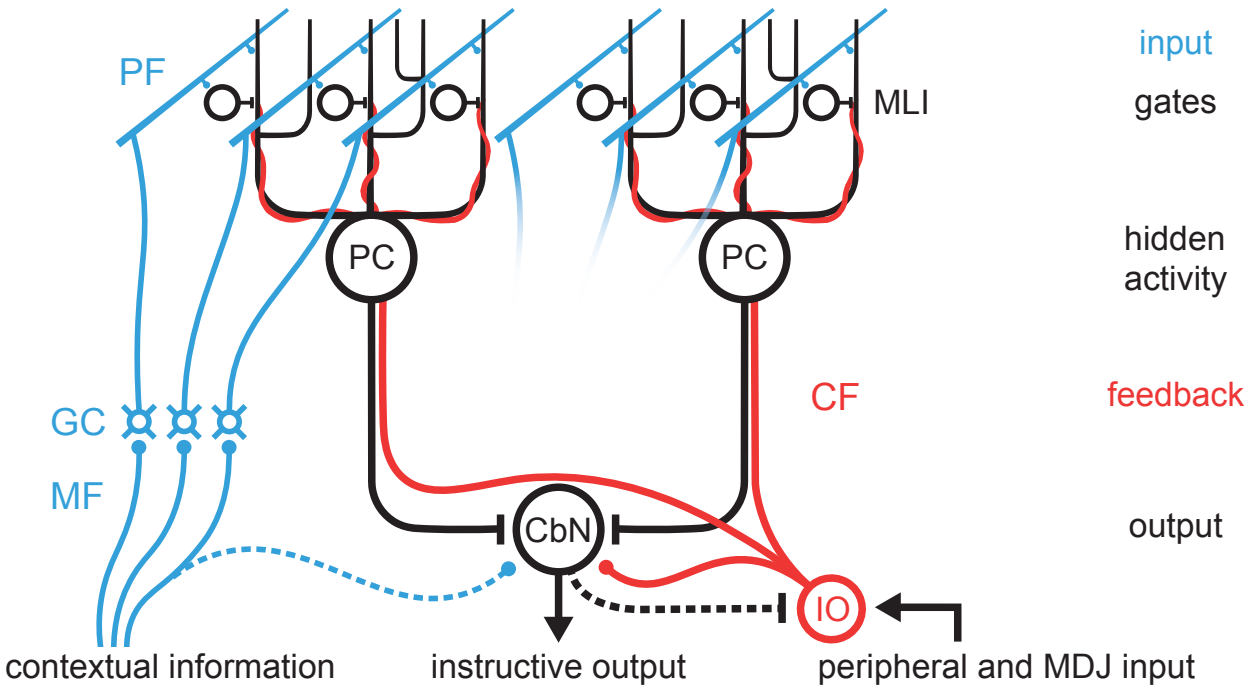

Figure 4: **The cerebellum as a two layer DGN.** Contextual information from the mossy fiber (MF)/granule cell (GC) pathway is conveyed as input to the network via parallel fibers (PFs) that form synapses onto both the dendritic branches of Purkinje cells and molecular layer interneurons (MLIs). The inhibitory MLIs act as input-dependent gates of Purkinje cell dendritic branches. Purkinje cells converge onto the cerebellar nuclear neurons (CbNs) and constitute the output of the cerebellar network. The climbing fibers (CFs, red) originating in the inferior olive (IO) convey the feedback signal that is used to tune both the Purkinje cells, based on which inputs are gated on or off, and also the CbNs. Excitatory and inhibitory connections are depicted as round- and T-ends, respectively. Dashed lines represent connections not included in the model.

DGNs can learn inverse Kinematics

The cerebellum is thought to implement inverse motor control models [71, 72]. We therefore applied our proposed DGN network to the SARCOS benchmark [73], which is an inverse kinematics dataset collected using a 7 degree-of-freedom robot arm (Fig. 5). The goal is to learn an inverse model, and predict 7 joint torques given the joint positions, velocities, and accelerations for each of the 7 joints (corresponding to a 21 dimensional input).

The target output, r^*, is the desired torque, given the 21-dimensional input. There are seven joints, so we train seven different networks, each with its own target output. We use DGN networks with 20 Purkinje cells, each having 5000 branches, and minimize the quadratic loss (4).

In Fig. 5 we plot the target torques for each joint (dots) along with the predictions

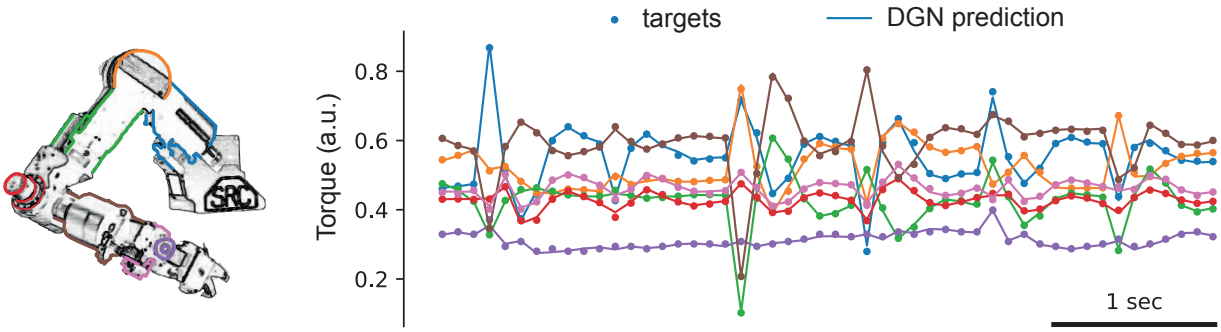

Figure 5: Sarcos experiment. DGNs can solve a challenging motor control task: predicting torques from the proprioceptive inputs. The data comes from a SARCOS dexterous robotic arm [73], pictured on the left. The inputs are position, velocity and acceleration of the joints (21 dimensional variables); the targets are the desired torques (7 dimensional). Example targets (normalized to keep the training data between 0 and 1) are shown with dots, the lines are the output of our network. Performance is very good; only rarely is there a visible difference between the dots and the lines.

of the DGN (lines; chosen for ease of comparison as there is no data between the points). The lines follow the points very closely, even when there are large fluctuations, indicating that the DGN is faithfully predicting torques. The performance of our network (mean squared error on test data in the original torque units) exceeds that of most machine learning algorithms (Supplementary Table S1) while using fewer (or an equal number of) samples to learn. This illustrates the power of DGNs; we now turn to a cerebellar task much more typical of computational and experimental neuroscience.

Vestibulo-ocular reflex, and adaptation to gain changes

When an animal moves its head, to maintain a stable image on the retina it moves its eyes in the opposite direction. This is known as the vestibulo-ocular reflex (VOR), and a key feature of it is that it is plastic: animals can adapt quickly when the relationship between the head movement and visual feedback is changed, as occurs as animals grow or are given corrective lenses. VOR gain adaptation relies critically on the cerebellum, and has been used to study cerebellar motor learning for decades [74–78].

We thus applied our DGN network to model learning of VOR gain changes. The gain, denoted G, is the ratio of the desired eye velocity to the head velocity (multiplied by -1 because the eyes and head move in opposite direction, to keep with the convention that the gain is reported as a positive number). When the gain is (artificially) changed, at first animals move their eyes at the wrong speed, but after about 15 minutes they learn to compensate [76, 77].

We trained our network on a head velocity signal of the form

$$s(t) = \sin(\omega_1 t) + \sin(\omega_2 t)\,, \tag{6}$$

with $\omega_1 = 13.333$ and $\omega_2 = 20.733$ (corresponding to 2.12 and 3.30 Hz, respectively).

This was chosen to mimic, approximately, the irregular head velocities encountered in natural viewing conditions. Following Clopath et. al. [79], we assumed that the Purkinje cells receive delayed versions of this signal. The ith input signal, $x_i(t)$, which arrives via the parallel fibers, is modelled as

$$x_i(t) = s(t - \tau_i)\,, \tag{7}$$

with delays, τ_i, spanning the range 50-300 ms. The cerebellum needs to compute the scaled version of the eye velocity: $r^*(t) = Gs(t)$ (as mentioned above, the actual eye movement is $-r^*(t)$, but we follow the standard convention). Learning was online, and we updated the weights every 500 ms, to approximately match the climbing fiber firing rate [80].

The DGN contained 20 Purkinje cells, with 10 branches each. As a baseline, we trained an MLP with the same number of weights (resulting in 200 hidden neurons). We used quadratic loss for both the DGN and the MLP and, as in [79], we assumed $n = 100$ parallel fibers and a single output. Each branch received input from all 100 parallel fibers. Gating (Eq. (2)) was controlled by $x_i(t)$ (given in Eq. (7)), refelecting the parallel fiber influence on molecular layer interneuronss (Fig. 4); see Methods for details. Given the timescale of the signal (2-3 Hz), any individual branch was gated on for about 500 ms at a time. The networks were pre-trained on a gain, G, of 1. We implemented four jump changes: first to 0.7, then back to 1.0, then to 1.3, and, finally, back to 1.0; in all cases, for 30 minutes (Fig. 6A).

Performance for both the DGN and the MLP were comparable and, after suitably adjusting the learning rates, the networks were able to learn in 15-20 minutes (Fig. 6A, B). Figure 6C shows the target and predicted head velocities immediately before and after each gain change. Not surprisingly, immediately after a gain change, the network produces output with the old gain.

Although both the DGN and the MLP solve this task, their internal mechanisms are remarkably different. Figure 6D shows the connection strengths between parallel fibers ($x_i(t)$, Eq. (7)) and Purkinje cells, after learning, as a function of the delay, τ_i. Every branch of the DGN (top panel; blue) develops a smooth connectivity pattern: parallel fibers that have similar delays have similar strengths. The smoothness of weights versus delay constitutes a strong prediction of our model.

2.3 Testing predictions of the DGN in behaving animals.

A critical feature of our model is that the gates (in the case of the cerebellum, the molecular layer interneurons) should remain stable over learning, or at least be more stable than other parts of the circuit, such as climbing fiber inputs to Purkinje cells. To test this, we performed simultaneous two-photon calcium imaging of molecular layer interneurons (MLIs) and Purkinje cell dendrites (Fig. 7A,B) in awake behaving head-fixed mice. Imaging occurred while the mouse was learning to make an association between a tone cue and a reward (Fig. 7C; note the absence of licks between the cue and the reward in the first 18 trials, before learning). To assess the stability of responses across learning, we computed a trial-wise population vector response to reward delivery (vector of mean response in 1 second following reward delivery for each

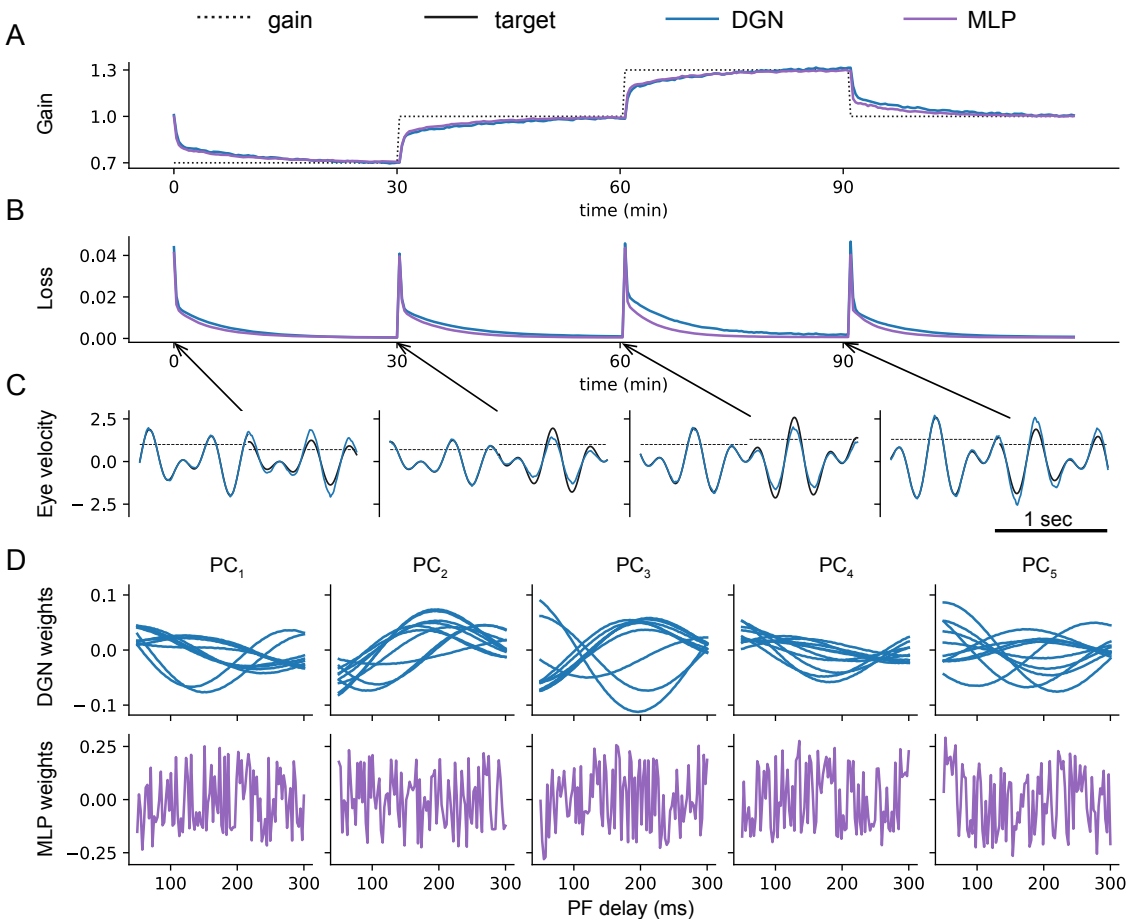

Figure 6: VOR adaptation task. We trained the networks on gain $G = 1$, then changed the gain every 30 minutes. Results are shown for the Dendritic Gated Network (DGN) and a multi-layer perceptron (MLP). **A**. Dashed lines are true gain versus time; blue and purple lines are gains computed by the DGN and MLP, respectively. For both networks, gains were inferred almost perfectly after 15-20 minutes. **B**. Performance, measured as mean squared error between the the true angular velocity, $Gs(t)$ (Eq. (6)), and the angular velocity inferred by the networks. Same color code as panel A. **C**. Comparison of target angular velocity versus time (black) to that predicted by the DGN (blue). (A plot for the MLP is similar.) Before the gain change, the two are almost identical; immediately after the gain change, the network uses the previous gain. **D**. Top panel: Parallel fiber weights for the DGN network versus delay, τ_i (Eq. (7)). Each panel shows 10 branches; 5 Purkinje cells are shown (chosen randomly out of 20). The weights vary smoothly with delay. Bottom panel: MLP weight profile, except that dendritic brances are replaced by the whole neuron (all 100 parallel fibers). For the MLP, the weights with similar delays are effectively uncorrelated.

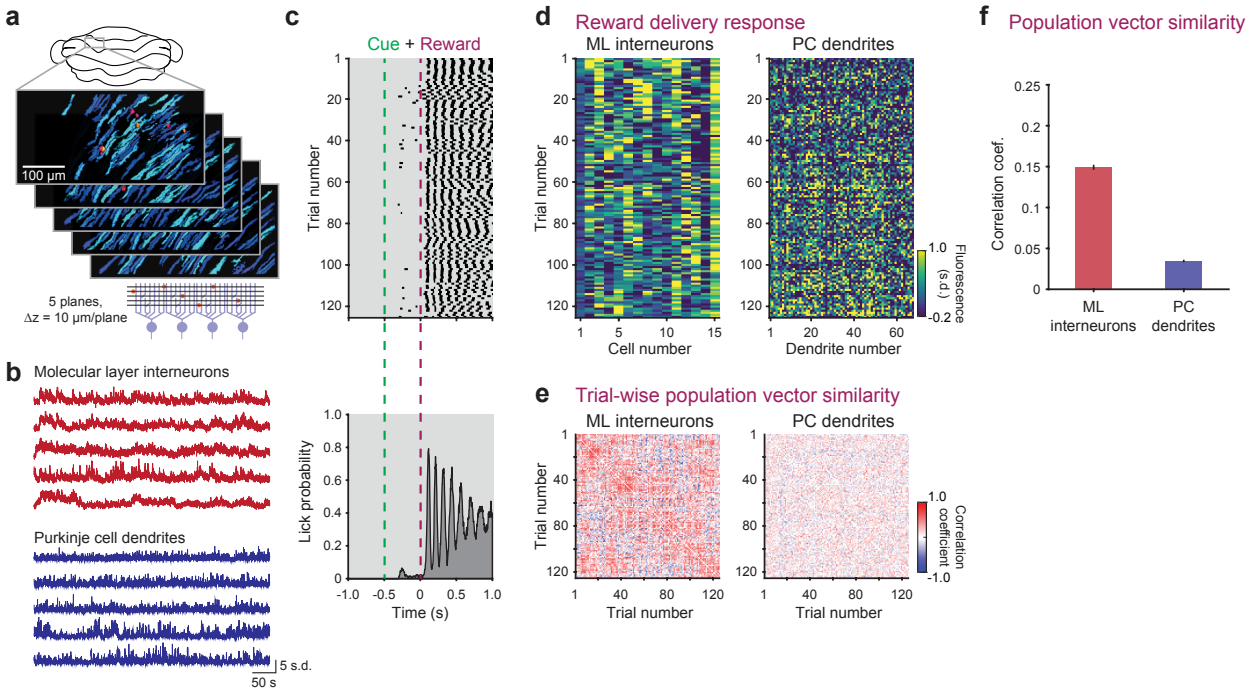

Figure 7: Testing experimental predictions of DGN during learning of cue-reward association. **A**. Simultaneous multi-plane 2-photon imaging of molecular layer interneurons (red hues) and Purkinje cell dendrites (blue hues) expressing GCaMP7f. Images were acquired across 5 planes at an effective rate of 9.7 Hz. **B**. Example traces of simultaneously recorded MLIs (red, top) and Purkinje cell dendrites (blue, bottom). **C**. Licking responses of mice during initial 125 trials of cue-reward pairing showing licking on individual trials (top) and mean lick probability (bottom). **D**. Trial-wise reward delivery responses in MLIs (left, n = 15) and Purkinje cell dendrites (right, n = 67) calculated as mean response in 1 s window after reward delivery. **E**. Similarity matrix of population vector response for MLIs (left) and Purkinje cell dendrites (right). **F**. Mean pairwise correlations of population vector responses in MLIs and Purkinje cell dendrites. MLIs responses exhibit greater trial-by-trial consistency. Data are shown as mean± S.E.M.

neuron; Fig. 7D). We compared the stability of these population response vectors in MLIs and Purkinje cell dendrites (reflecting climbing fiber input) over the course of the first 125 cue-reward pairing trials (Fig. 7E). The MLI population vector response was significantly more stable across these learning trials than the corresponding population response vector in Purkinje cell dendrites (Fig. 7F), consistent with the tendency for DGN gates to remain stable while other elements evolve with learning.

3 Discussion

A critical open question in neuroscience is: what learning rules ensure that synaptic strengths are updated in a way that improves performance? Answering this is difficult

in large part because of the way we think about computation, which is that networks map input to output in stages, with the input gradually transformed, until eventually, in the output layer, the relevant features are easy to extract. There is certainly some evidence for this. It is, for example, much harder to extract which face a person is looking at from activity in visual area V1 than in fusiform face area [81, 82]. While this strategy for computing is reasonable, it has a downside: the relationship between activity in intermediate layers and activity in the output layer is highly nontrivial, which makes it especially hard for the brain to determine how weights in intermediate layers should change.

Here we propose that the brain might take a different approach, one based on Dendritic Gated Networks, or DGNs, which is a variant of the Gated Linear Network [1, 2]. With this architecture, each neuron is active for a relatively small region of the input space; for the rest, it is gated "off". Each neuron receives its input from the layer below, as in conventional networks, but its goal is not to transform that input; instead, its goal is to predict the output of the whole network. That makes the role of every neuron transparent (all neurons in all layers are doing the same thing), which makes learning simple – all that is required is a delta rule.

The ease of learning makes DGNs strong candidates for biological networks. In addition, we showed they are compatible with the architecture and function of the cerebellum, and that they perform well on three nontrivial tasks. Finally, we supplied preliminary experimental support for gating, which in the cerebellum we hypothesize is done by the molecular layer interneurons.

DGNs make three strong predictions for the cerebellum. First, the activity of the molecular layer interneurons should depend solely on parallel fiber input and should not change with learning – or change very slowly relative to the timescale over which Purkinje cells learn, the latter measured in single trials [83]. This prediction is consistent with our *in vivo* imaging experiments. Second, dendritic branches should be in one of two states, determined by molecular layer interneuron activity: either a branch receives very little MLI input, so that it can transmit information from parallel fibers to Purkinje cells, or it receives very large MLI input, so that it cannot transmit information. Testing the second prediction is challenging, but could be addressed using a combination of cellular resolution all-optical stimulation and voltage imaging, a technical feat that may soon be within reach [84, 85]. Third, for parallel fibers carrying delayed information about head position, the parallel fiber to Purkinje cell weights should be a smooth function of the delay (Fig. 6d, top panel).

In summary, Dendritic Gated Networks are strong candidates for biological networks – and not just in the cerebellum; they could be used anywhere there is approximately feedforward structure. They come with two desirable features: biologically plausible learning, and rapid, data-efficient learning. And they imply a novel role for inhibitory neurons, which is that they are used for gating dendritic branches on and off. Importantly, they make strong, experimentally testable, predictions, so we will soon know whether they are actually used in the brain.

4 Methods

4.1 Model

The network we use in our model is described in Eqs. (1) and (2), and the learning rules are given in Eq. (3). In particular, Eq. (5) is used in all our simulations except for MNIST, where the output is categorical. In that case, we bound neural activities so they can represent probabilities. We use a standard sigmoid function, $\sigma(z) = e^z/(1 + e^z)$, albeit modified slightly,

$$\phi(z) = \textsc{clip}_{\epsilon}^{1-\epsilon}\big(\sigma(z)\big) \tag{8}$$

where $\textsc{clip}_a^b(\cdot)$ clips values between a and b (so the right hand side is zero if $\sigma(z)$ is smaller than ϵ or larger than $1 - \epsilon$). Clipping is used for bounding the loss as well as the gradients; this helps with numerical stability, and also enables a worst-case regret analysis [1, 2]. We set ϵ to 0.01, so neural activity lies between values 0.01 and .99.

The loss of neuron i in layer k in this case is given by

$$\ell(r^*, r_{k,i}) = -r^* \log r_{k,i} - (1 - r^*) \log \big(1 - r_{k,i}\big) \,. \tag{9}$$

Consequently, the update rule for the weights, Eq. (3), is (after a small amount of algebra)

$$\Delta w_{k,ij}^b = \eta g_{k,i}^b(\mathbf{x}) \mathbb{1}(\epsilon < r_{k,i} < 1 - \epsilon) \big(r^* - r_{k,i}\big) h_{k-1,j} \tag{10}$$

where $\mathbb{1}(\cdot)$ is 1 when its argument is true and 0 otherwise. The fact that the learning is zero when $r_{k,i}$ is outside the range $[\epsilon, 1 - \epsilon]$ follows because $d\phi(z)/dz = 0$ when z is outside this range (see Eq. (8)). This ensures that learning saturates when weights become too large (either positive or negative). However, this can cause problems if the output is very wrong: when $r^* = 1$ and $r_{k,i} < \epsilon$ or $r^* = 0$ and $r_{k,i} > 1 - \epsilon$. To address this, we allow learning in this regime. We can do this compactly by changing the learning rule to

$$\Delta w_{k,ij}^b = \eta g_{k,i}^b(\mathbf{x}) \mathbb{1}(|r^* - r_{k,i}| > \epsilon) \big(r^* - r_{k,i}\big) h_{k-1,j} \,. \tag{11}$$

Essentially, this rule says: stop learning when $r_{k,i}$ is within ϵ of r^*. See [86] for a complementary view of how categorical problems might be solved by gated neurons in the brain.

For a compact summary of the equations (given as pseudocode), see Supplementary Algorithms S1 and S2.

4.2 Simulations

Simulations were written using JAX [87], the DeepMind JAX Ecosystem [88], and Colab [89].

Catastrophic Forgetting. We adopt the pixel-permuted MNIST benchmark [55, 58], which is a sequence of MNIST digit classification tasks with different pixel permutations. Each task consists of 60,000 training images and 10,000 test images, all images are deskewed. Models are trained sequentially across 10 tasks, performing a single pass over each. We provide the implementation details of the baselines below. We display the parameters swept during grid search in Supplementary Table S2.

DGN. We use networks composed of 100 and 20 units in the hidden layers and a single linear neuron layer for the output. Each neuron in the hidden layer has 10 dendritic branches. The output of the network is determined by the last neuron. MNIST has 10 classes, each corresponding to a digit. Therefore, we utilize 10 DGN networks, each encoding the probability of a distinct class. Each of these networks are updated during training using a learning rate of 10^{-2}. During testing, the class with the maximum probability is chosen. Images are scaled and shifted so that the input range is $[-1, 1]$. The gating vectors, $\mathbf{v}^b_{k,i}$, are chosen randomly on the unit sphere, which can be achieved by sampling from an isotropic Normal distribution and then dividing by the L2 norm. The biases, $\theta^b_{k,i}$ are drawn independently from a centred normal distribution with standard deviation 0.05.

MLP and EWC. We use a ReLu network with 1000 and 200 neurons in the hidden layers and 10 linear output units with cross entropy loss. In this setting, the MLP and EWC have the same number of neurons as DGN but fewer plastic weights in total. We use the ADAM optimization method [90] with a learning rate of 10^{-4} (see Supplementary Table S2 for details of the hyperparameter optimization), in conjunction with dropout. We use mini-batches of 20 data points. For EWC, we draw 100 samples for computing the Fisher matrix diagonals and set the regularization constant to 10^3.

Inverse Kinematics. Each DGN network has 20 Purkinje cells with 5000 branches each. We use a quadratic loss (4) with a learning rate $\eta = 10^{-5}$ for 2000 epochs (2000 passes over the dataset). The inputs are centered at 0 and scaled to unit variance per dimension, the targets are scaled so that they lie between 0 and 1. The reported MSEs are computed on the test set based on inverse transformed predictions (thus undoing the target scaling). The gating parameters are chosen in the same way as for the Mnist simulations (see above).

We discovered that the the training set of the SARCOS dataset (downloaded from `http://www.gaussianprocess.org/gpml/data/` on 15 December 2020) includes test instances. To the best of our knowledge, other recent studies using the SARCOS dataset [91, 92] reported results with this train/test setting. This means that the reported errors are measures of capacity rather than generalization. We compare the performance of DGN against the best known SARCOS results in Supplementary Table S1 using the existing train/test split. If we exclude the test instances from the train set, we get an MSE for the DGN of 0.84 using the same network setting and parameters.

VOR. The gating parameters $v^b_{k,ij}$ and $\theta^b_{k,i}$ (Eq. (2)), were drawn independently from the standard normal distribution. Learning rate was $\eta = 10^{-5}$ for DGN and $\eta = 0.02$ for MLP.

4.3 Animal experiments

Animal housing and surgery All animal procedures were approved by the local Animal Welfare and Ethical Review Board and performed under license from the UK Home Office in accordance with the Animals (Scientific Procedures) Act 1986 and generally followed procedures described previously [93]. Briefly, we used PV-Cre mice (B6;129P2-Pvalbtm1(cre)Arbr/J) [94] crossed with C57/BL6 wild type mice. Mice were group housed before and after surgery and maintained on a 12:12 day-night cycle. Surgical procedures were identical to those described in [93], except that we injected Cre-dependent GCaMP7f (pGP-AAV-CAG-FLEX-jGCaMP7f-WPRE [serotype 1]; [95]) diluted from its stock titer at 1:25. After mice had recovered from surgery, they were placed under water restriction for at least 5 days during which time they were acclimated to the recording setup and expression-checked. All mice were maintained at 80-85 percent of their initial weight over the course of imaging. Trained mice typically received all their water for the day from reward during the behavioral task, while naïve mice were supplemented to 1 g water per day with Hydrogel.

Cue-reward association training Mice were trained on a conditioning protocol in which an auditory cue (4 kHz, 100 ms duration) was paired with a reward delivered 500 ms after cue onset, similar to the conditioning paradigm described in [93]. Responses of MLIs and PC dendrites to reward delivery were recorded and analyzed during the first 125 trials after initial cue-reward pairing to assess response consistency across the initial learning phase of this association.

Two-photon calcium imaging, data acquisition, and processing Imaging experiments were performed using a 16x/0.8 NA objective (Nikon) mounted on a Sutter MOM microscope equipped with the Resonant Scan Box module. A Ti:Sapphire laser tuned to 930 nm (Mai Tai, Spectra Physics) was raster scanned using a resonant scanning galvanometer (8 kHz, Cambridge Technologies) and images were collected at 512x256 pixel resolution over fields of view of 450x225 μm per plane. Volumetric imaging across 5 planes spaced by 10 μm (depth ranging 25-65 μm below pial surface) were performed using a P-726 PIFOC High-Load Objective Scanner (Physik Instruments) at an effective volume rate of 9.7 Hz. The microscope was controlled using ScanImage (Version 2015, Vidrio Technologies) and tilted to 10 degrees such that the objective was orthogonal to the surface of the brain and coverglass. ROIs corresponding to single MLIs and PC dendrites were extracted using a combination of Suite2p software [96] for initial source extraction and custom-written software to merge PC dendritic ROIs across recording planes, which exhibited highly correlated calcium signals. Calcium signals corresponding to individual MLI somata and PC dendrites, which were easily distinguishable based on their shape, were computed as (F-F0)/F0 where F was the signal measured at each point in time and F0 is the 8th percentile of a 200 second rolling average surrounding each data time point). A neuropil correction coefficient of 0.5 (50 percent of neuropil signal output from Suite2p) was applied to all ROIs. A range of baseline durations and neuropil correction coefficients were tested and varying these parameters did not alter the main findings. Fluorescence changes for each neuron were then z-scored over time to facilitate comparisons between individual neurons with

different baseline expression levels. Behavioural events and imaging synchronization signals were acquired using PackIO (see [93] for detailed description) and aligned offline using custom written scripts.

Code availability. We provide pseudo code in Supplementary Algorithms S1 andS2. A simple python implementation can be accessed via `https://github.com/deepmind/deepmind-research/blob/master/gated_linear_networks/colabs/dendritic_gated_network.ipynb`.

Data availability. The data that support the findings of this study are available from the corresponding authors upon reasonable request. Additional analysis made use of standard publicly available benchmarks including MNIST [97] and SARCOS (`http://www.gaussianprocess.org/gpml/data/`).

Acknowledgements. We thank Timothy Lillicrap, Gregory Wayne, and Eszter Vértes for valuable feedback. Michael Häusser is supported by the Wellcome Trust and the European Research Council. Peter Latham is supported by the Gatsby Charitable Foundation and the Wellcome Trust.

Author contributions. ES and AGB developed the computational model with advice from JV, CC, PEL, DB, and MHut. AGB, ES, and SK performed simulation experiments and analysis with advice from CC and PEL. DK and MBea acquired and analyzed neuronal data with advice from MHäu. PEL, AGB, ES, and DK wrote the paper with help from all other authors. AGB and ES managed the project with support from MBot, CC, JV, and DB.

Competing Interests. The authors declare no competing interests.

References

[1] Veness, J. *et al.* Online learning with gated linear networks. *arXiv preprint arXiv:1712.01897* (2017).

[2] Veness, J. *et al.* Gated linear networks. *Proceedings of the AAAI Conference on Artificial Intelligence (To Appear)* (2021).

[3] Burke, C. J. *et al.* Layered reward signalling through octopamine and dopamine in drosophila. *Nature* **492**, 433–437 (2012).

[4] Liu, C. *et al.* A subset of dopamine neurons signals reward for odour memory in drosophila. *Nature* **488**, 512–516 (2012).

[5] Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks* **61**, 85–117 (2015).

[6] Grossberg, S. Competitive learning: From interactive activation to adaptive resonance. *Cogn. Sci.* **11**, 23–63 (1987).

[7] Crick, F. The recent excitement about neural networks. *Nature* **337**, 129–132 (1989).

[8] Roelfsema, P. R. & Holtmaat, A. Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience* **19**, 166 (2018).

[9] Whittington, J. C. & Bogacz, R. Theories of error back-propagation in the brain. *Trends in cognitive sciences* **23**, 235–250 (2019).

[10] Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J. & Hinton, G. Backpropagation and the brain. *Nature Reviews Neuroscience* **21**, 335–346 (2020).

[11] Kelley, H. J. Gradient theory of optimal flight paths. *Ars Journal* **30**, 947–954 (1960).

[12] Linnainmaa, S. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki* 6–7 (1970).

[13] Rumelhart, D. E., , G. E. & Williams, R. J. Learning representations by back-propagating errors. In Anderson, J. A. & Rosenfeld, E. (eds.) *Neurocomputing: Foundations of Research*, 696–699 (MIT Press, Cambridge, MA, USA, 1988).

[14] French, R. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* **3**, 128–135 (1999).

[15] Herzfeld, D. J., Vaswani, P. A., Marko, M. K. & Shadmehr, R. A memory of errors in sensorimotor learning. *Science* **345**, 1349–1353 (2014).

[16] Wolpert, D. M. & Flanagan, J. R. Computations underlying sensorimotor learning. *Current opinion in neurobiology* **37**, 7–11 (2016).

[17] Lillicrap, T. P., Cownden, D., Tweed, D. B. & Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications* **7**, 13276 (2016). URL `http://www.nature.com/articles/ncomms13276`.

[18] Akrout, M., Wilson, C., Humphreys, P., Lillicrap, T. & Tweed, D. B. Deep learning without weight transport. In *Advances in Neural Information Processing Systems*, vol. 6, e22901 (2017).

[19] Guerguiev, J., Lillicrap, T. P. & Richards, B. A. Towards deep learning with segregated dendrites. *eLife* **6**, e22901 (2019).

[20] Sacramento, J., Costa, R. P., Bengio, Y. & Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems*, 8721–8732 (2018).

[21] Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A. & Naud, R. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *bioRxiv* (2020). URL `https://www.biorxiv.org/content/early/2020/03/31/2020.03.30.015511`.

[22] Samadi, A., Lillicrap, T. P. & Tweed, D. B. Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural Computation* **29**, 578–602 (2017).

[23] Belilovsky, E., Eickenberg, M. & Oyallon, E. Greedy layerwise learning can scale to ImageNet. In *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 583–593 (PMLR, 2019). URL `http://proceedings.mlr.press/v97/belilovsky19a.html`.

[24] Nøkland, A. Direct feedback alignment provides learning in deep neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, 1045–1053 (Curran Associates Inc., Red Hook, NY, USA, 2016).

[25] Nøkland, A. & Eidnes, L. H. Training Neural Networks with Local Error Signals. In *International Conference on Machine Learning*, 4839–4850 (PMLR, 2019). URL `http://proceedings.mlr.press/v97/nokland19a.html`. ISSN: 2640-3498.

[26] Löwe, S., O'Connor, P. & Veeling, B. Putting an end to end-to-end: Gradient-isolated learning of representations. In *Advances in Neural Information Processing Systems*, 3033–3045 (2019).

[27] Pogodin, R. & Latham, P. E. Kernelized information bottleneck leads to biologically plausible 3-factor hebbian learning in deep networks. *arXiv preprint arXiv:2006.07123* (2020). `2006.07123`.

[28] Podlaski, W. F. & Machens, C. K. Biological credit assignment through dynamic inversion of feedforward networks (2020). `2007.05112`.

[29] Golkar, S., Lipshutz, D., Bahroun, Y., Sengupta, A. M. & Chklovskii, D. B. A biologically plausible neural network for local supervision in cortical microcircuits (2020). `2011.15031`.

[30] Budden, D. *et al.* Gaussian gated linear networks. In *Advances in Neural Information Processing Systems* (2020).

[31] Sezener, E., Hutter, M., Budden, D., Wang, J. & Veness, J. Online learning in contextual bandits using gated linear networks. In *Advances in Neural Information Processing Systems* (2020).

[32] Wang, J., Sezener, E., Budden, D., Mutter, M. & Veness, J. A combinatorial perspective on transfer learning. In *Advances in Neural Information Processing Systems* (2020).

[33] Widrow, B. & Hoff, M. E. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, 96–104 (IRE, New York, 1960).

[34] Llinás, R. & Sugimori, M. Electrophysiological properties of in vitro purkinje cell somata in mammalian cerebellar slices. *The Journal of physiology* **305**, 171–195 (1980).

[35] Walter, J. T. & Khodakhah, K. The linear computational algorithm of cerebellar purkinje cells. *Journal of Neuroscience* **26**, 12861–12872 (2006).

[36] Chen, S., Augustine, G. J. & Chadderton, P. Serial processing of kinematic signals by cerebellar circuitry during voluntary whisking. *Nature communications* **8**, 1–13 (2017).

[37] Eccles, J. Functional meaning of the patterns of synaptic connections in the cerebellum. *Perspectives in biology and medicine* **8**, 289–310 (1965).

[38] Dizon, M. J. & Khodakhah, K. The role of interneurons in shaping purkinje cell responses in the cerebellar cortex. *Journal of Neuroscience* **31**, 10463–10473 (2011).

[39] Brown, A. M. *et al.* Molecular layer interneurons shape the spike activity of cerebellar purkinje cells. *Scientific reports* **9**, 1–19 (2019).

[40] Andersen, P., Eccles, J. & Voorhoeve, P. Postsynaptic inhibition of cerebellar purkinje cells. *Journal of neurophysiology* **27**, 1138–1153 (1964).

[41] Mittmann, W., Koch, U. & Häusser, M. Feed-forward inhibition shapes the spike output of cerebellar purkinje cells. *The Journal of physiology* **563**, 369–378 (2005).

[42] Häusser, M. & Clark, B. A. Tonic synaptic inhibition modulates neuronal output pattern and spatiotemporal synaptic integration. *Neuron* **19**, 665–678 (1997).

[43] Oldfield, C. S., Marty, A. & Stell, B. M. Interneurons of the cerebellar cortex toggle purkinje cells between up and down states. *Proceedings of the National Academy of Sciences* **107**, 13153–13158 (2010).

[44] Callaway, J. C., Lasser-Ross, N. & Ross, W. N. Ipsps strongly inhibit climbing fiber-activated [ca2+] i increases in the dendrites of cerebellar purkinje neurons. *Journal of Neuroscience* **15**, 2777–2787 (1995).

[45] Wulff, P. *et al.* Synaptic inhibition of purkinje cells mediates consolidation of vestibulo-cerebellar motor learning. *Nature neuroscience* **12**, 1042–1049 (2009).

[46] Gaffield, M. A., Rowan, M. J., Amat, S. B., Hirai, H. & Christie, J. M. Inhibition gates supralinear ca2+ signaling in purkinje cell dendrites during practiced movements. *Elife* **7**, e36246 (2018).

[47] Carpenter, G. A. & Grossberg, S. The art of adaptive pattern recognition by a self-organizing neural network. *Computer* **21**, 77–88 (1988).

[48] McCloskey, M. & Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation* **24**, 109 – 165 (1989). URL `http://www.sciencedirect.com/science/article/pii/S0079742108605368`.

[49] Robins, A. V. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connect. Sci.* **7**, 123–146 (1995).

[50] Caruana, R. Multitask learning. *Machine Learning* **28**, 41–75 (1997). URL `https://doi.org/10.1023/A:1007379606734`.

[51] Rebuffi, S.-A., Kolesnikov, A., Sperl, G. & Lampert, C. H. icarl: Incremental classifier and representation learning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017). URL `http://dx.doi.org/10.1109/CVPR.2017.587`.

[52] Donahue, J. *et al.* Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR* **abs/1310.1531** (2013). URL `http://arxiv.org/abs/1310.1531`. `1310.1531`.

[53] Razavian, A. S., Azizpour, H., Sullivan, J. & Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2014). URL `http://dx.doi.org/10.1109/CVPRW.2014.131`.

[54] Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014). URL `http://dx.doi.org/10.1109/CVPR.2014.81`.

[55] Kirkpatrick, J. *et al.* Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* **114**, 3521–3526 (2017). URL `https://www.pnas.org/content/114/13/3521`. `https://www.pnas.org/content/114/13/3521.full.pdf`.

[56] Zenke, F., Poole, B. & Ganguli, S. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 3987–3995 (JMLR.org, 2017).

[57] Schwarz, J. *et al.* Progress &; compress: A scalable framework for continual learning. In Dy, J. & Krause, A. (eds.) *Proceedings of the 35th International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, 4528–4537 (PMLR, Stockholmsmässan, Stockholm Sweden, 2018). URL `http://proceedings.mlr.press/v80/schwarz18a.html`.

[58] Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A. & Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks (2013). `1312.6211`.

[59] Marr, D. A theory of cerebellar cortex. *The Journal of Physiology* **202**, 437–470.1 (1969). URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1351491/`.

[60] Albus, J. S. A theory of cerebellar function. *Mathematical Biosciences* **10**, 25–61 (1971). URL `http://www.sciencedirect.com/science/article/pii/0025556471900514`.

[61] Ito, M. & Kano, M. Long-lasting depression of parallel fiber-purkinje cell transmission induced by conjunctive stimulation of parallel fibers and climbing fibers in the cerebellar cortex. *Neuroscience Letters* **33**, 253 – 258 (1982). URL `http://www.sciencedirect.com/science/article/pii/0304394082903809`.

[62] Narain, D., Remington, E. D., Zeeuw, C. I. D. & Jazayeri, M. A cerebellar mechanism for learning prior distributions of time intervals. *Nature Communications* **9**, 469 (2018).

[63] Shadmehr, R. Population coding in the cerebellum and its implications for learning from error. *bioRxiv* (2020).

[64] Pedroarena, C. M. & Schwarz, C. Efficacy and short-term plasticity at gabaergic synapses between purkinje and cerebellar nuclei neurons. *Journal of Neurophysiology* **89**, 704–715 (2003).

[65] Pedroarena, C. M. Short-term plasticity at purkinje to deep cerebellar nuclear neuron synapses supports a slow gain-control mechanism enabling scaled linear encoding over second-long time windows. *bioRxiv* 749259 (2019).

[66] Morishita, W. & Sastry, B. Postsynaptic mechanisms underlying long-term depression of gabaergic transmission in neurons of the deep cerebellar nuclei. *Journal of Neurophysiology* **76**, 59–68 (1996).

[67] Aizenman, C. D., Manis, P. B. & Linden, D. J. Polarity of long-term synaptic gain change is related to postsynaptic spike firing at a cerebellar inhibitory synapse. *Neuron* **21**, 827–835 (1998).

[68] Aizenman, C. D. & Linden, D. J. Rapid, synaptically driven increases in the intrinsic excitability of cerebellar deep nuclear neurons. *Nature neuroscience* **3**, 109–111 (2000).

[69] Zheng, N. & Raman, I. M. Synaptic inhibition, excitation, and plasticity in neurons of the cerebellar nuclei. *The Cerebellum* **9**, 56–66 (2010).

[70] Hull, C. Prediction signals in the cerebellum: beyond supervised motor learning. *Elife* **9**, e54073 (2020).

[71] Kawato, M., Furukawa, K. & Suzuki, R. A hierarchical neural-network model for control and learning of voluntary movement. *Biological cybernetics* **57**, 169–185 (1987).

[72] Wolpert, D. M., Miall, R. C. & Kawato, M. Internal models in the cerebellum. *Trends in cognitive sciences* **2**, 338–347 (1998).

[73] Vijayakumar, S. & Schaal, S. Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, vol. 1, 288–293 (2000).

[74] Robinson, D. Adaptive gain control of vestibuloocular reflex by the cerebellum. *Journal of neurophysiology* **39**, 954–969 (1976).

[75] Miles, F. & Lisberger, S. Plasticity in the vestibulo-ocular reflex: a new hypothesis. *Annual review of neuroscience* **4**, 273–299 (1981).

[76] Lac, S., Raymond, J. L., Sejnowski, T. J. & Lisberger, S. G. Learning and memory in the vestibulo-ocular reflex. *Annual review of neuroscience* **18**, 409–441 (1995).

[77] Ito, M. Cerebellar learning in the vestibulo-ocular reflex. *Trends in cognitive sciences* **2**, 313–321 (1998).

[78] Boyden, E. S., Katoh, A. & Raymond, J. L. Cerebellum-dependent learning: the role of multiple plasticity mechanisms. *Annual review of neuroscience* **27** (2004).

[79] Clopath, C., Badura, A., De Zeeuw, C. I. & Brunel, N. A cerebellar learning model of vestibulo-ocular reflex adaptation in wild-type and mutant mice. *Journal of Neuroscience* **34**, 7203–7215 (2014).

[80] Armstrong, D. & Rawson, J. Activity patterns of cerebellar cortical neurones and climbing fibre afferents in the awake cat. *The Journal of Physiology* **289**, 425–448 (1979).

[81] Tsao, D. Y., Freiwald, W. A., Knutsen, T. A., Mandeville, J. B. & Tootell, R. B. Faces and objects in macaque cerebral cortex. *Nature neuroscience* **6**, 989–995 (2003).

[82] Zhen, Z., Fang, H. & Liu, J. The hierarchical brain network for face recognition. *PloS one* **8**, e59886 (2013).

[83] Yang, Y. & Lisberger, S. G. Purkinje-cell plasticity and cerebellar motor learning are graded by complex-spike duration. *Nature* **510**, 529–532 (2014).

[84] Packer, A. M., Russell, L. E., Dalgleish, H. W. & Häusser, M. Simultaneous all-optical manipulation and recording of neural circuit activity with cellular resolution in vivo. *Nature methods* **12**, 140–146 (2015).

[85] Roome, C. J. & Kuhn, B. Dendritic coincidence detection in purkinje neurons of awake mice. *Elife* **9**, e59619 (2020).

[86] Chung, S. & Abbott, L. Distributed local learning in context-switched linear networks (2021).

[87] Bradbury, J. *et al.* JAX: composable transformations of Python+NumPy programs (2018). URL `http://github.com/google/jax`.

[88] Babuschkin, I. *et al.* The DeepMind JAX Ecosystem (2020). URL `http://github.com/deepmind`.

[89] Ekaba, B. Google colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA. (2019). URL `https://doi.org/10.1007/978-1-4842-4470-8_7`.

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

[91] Tanno, R., Arulkumaran, K., Alexander, D., Criminisi, A. & Nori, A. Adaptive neural trees. In Chaudhuri, K. & Salakhutdinov, R. (eds.) *Proceedings of the 36th International Conference on Machine Learning*, vol. 97 of *Proceedings of Machine Learning Research*, 6166–6175 (PMLR, Long Beach, California, USA, 2019). URL `http://proceedings.mlr.press/v97/tanno19a.html`.

[92] Arik, S. O. & Pfister, T. Tabnet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence (To Appear)* (2021).

[93] Kostadinov, D., Beau, M., Pozo, M. B. & Häusser, M. Predictive and reactive reward signals conveyed by climbing fiber inputs to cerebellar purkinje cells. *Nature Neuroscience* **22**, 950–962 (2019).

[94] Hippenmeyer, S. *et al.* A developmental switch in the response of drg neurons to ets transcription factor signaling. *PLoS biol* **3**, e159 (2005).

[95] Dana, H. *et al.* High-performance calcium sensors for imaging activity in neuronal populations and microcompartments. *Nature Methods* **16**, 649–657 (2019).

[96] Pachitariu, M. *et al.* Suite2p: beyond 10,000 neurons with standard two-photon microscopy. *Biorxiv* (2017).

[97] LeCun, Y., Cortes, C. & Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* **2** (2010).

[98] Boyd, S. & Vandenberghe, L. *Convex Optimization* (Cambridge University Press, 2004). URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike-20&path=ASIN/0521833787`.

Supplementary Methods

Convexity

If we ignore clipping, which has no effect on the convexity proof, the structure of the loss ℓ as a function of the weight vector $\mathbf{w}$ is as follows: $\ell(r^*, r)$ with $r = \phi(h)$ and $h = \mathbf{c} \cdot \mathbf{w}$. Concretely, for neuron i in layer k, we have $r = r_{k,i}$ and $h = h_{k,i} \in \mathbb{R}$ and $\mathbf{w} = w^{\bullet}_{k,i\bullet} \in \mathbb{R}^{(n_{k-1}+1)\times B_{k,i}}$ and $\mathbf{c} = g^{\bullet}_{k,i}(\mathbf{x}) h_{k-1,\bullet} \in \mathbb{R}^{(n_{k-1}+1)\times B_{k,i}}$ and $\cdot$ denotes sum over j and b. If $\ell(r^*, \phi(h)))$ is convex in h, then ℓ is also convex in $\mathbf{w}$, since h is a linear function of $\mathbf{w}$ (e.g. [98] Sec.3.2.2). For quadratic loss (4) and ϕ being the identity, $\ell(r^*, \phi(h))) = \frac{1}{2}(r^* - h)^2$ is obviously convex in h hence $\mathbf{w}$. For log-loss (9) and $\phi(h) = \sigma(h) = 1/(1 + e^{-h})$, it is easy to show that $\partial^2 \ell(r^*, \phi(h)))/\partial h^2 = \sigma(h)(1 - \sigma(h)) > 0$, hence, again, ℓ is convex in h and therefore also in $\mathbf{w}$.

Inverse Kinematics

In Table S1 we compare the mean square error (MSE) obtained by DGN against baselines obtained from [30, 91, 92]. Note that, as mentioned in Methods, we (like others) used a test set that contained training examples.

Algorithm	MSE	Epochs
DGN	**0.002**	2000
Random forest	2.39	-
MLP	2.13	-
Stochastic decision tree	2.11	-
Gradient boosted tree	1.44	-
TabNet-S	1.25	55000
Adaptive neural tree	1.23	-
TabNet-M	0.28	55000
TabNet-L	0.14	55000
Gaussian Gated Linear Network	0.10	2000

Table S1: Test MSE and the number of passes over the dataset (ie, epochs) for DGN versus previously published methods on the SARCOS inverse dynamics dataset [73, 91, 92]. DGN obtains the best result, by a factor of 50.

Catastrophic Forgetting (permuted MNIST)

Hyerparameter selection. We select the hyperparameters for the three methods utilizing a grid search. The swept and the chosen parameters are displayed in Table S2.

Learning curves. In Fig. S1 we display the test performance of previously learned tasks (columns) as a function of the training across multiple tasks. To reduce clutter, a

Model	learning rate	dropout	regularization const
DGN	10^{-4}, 10^{-3}, $\mathbf{10^{-2}}$, 10^{-1}	–	–
MLP	10^{-6}, 10^{-5}, $\mathbf{10^{-4}}$, 10^{-3}	Yes, **No**	–
EWC	10^{-6}, 10^{-5}, $\mathbf{10^{-4}}$, 10^{-3}	Yes, **No**	10^{2}, $\mathbf{10^{3}}$, 10^{4}

Table S2: Parameters swept during grid search. The best parameters (shown in bold) are the ones that maximize the average test accuracy over 20 random seeds.

subset of the tasks (1, 2, 4, and 8, out of 10) are shown. The top left plot (train and test on task 1) shows that DGNs learns the first task much faster than all other methods. The plots to the right of that show retention on task 1 while the network is sequentially trained on subsequent tasks. MLP performances drop drastically after learning a few new tasks, while DGN and EWC show little forgetting. This is a remarkable feat for DGNs, which have no access to task boundaries and no explicit memory of previously learned tasks. EWC, on the other hand, has both. If we look at the four diagonal plots, we see that DGN learns new tasks faster than all other methods, although the difference gets smaller as more tasks are learned.

The final accuracies across the diagonal correspond to the left panel of Figure 3 whereas the final accuracies across the first row correspond to the right panel.

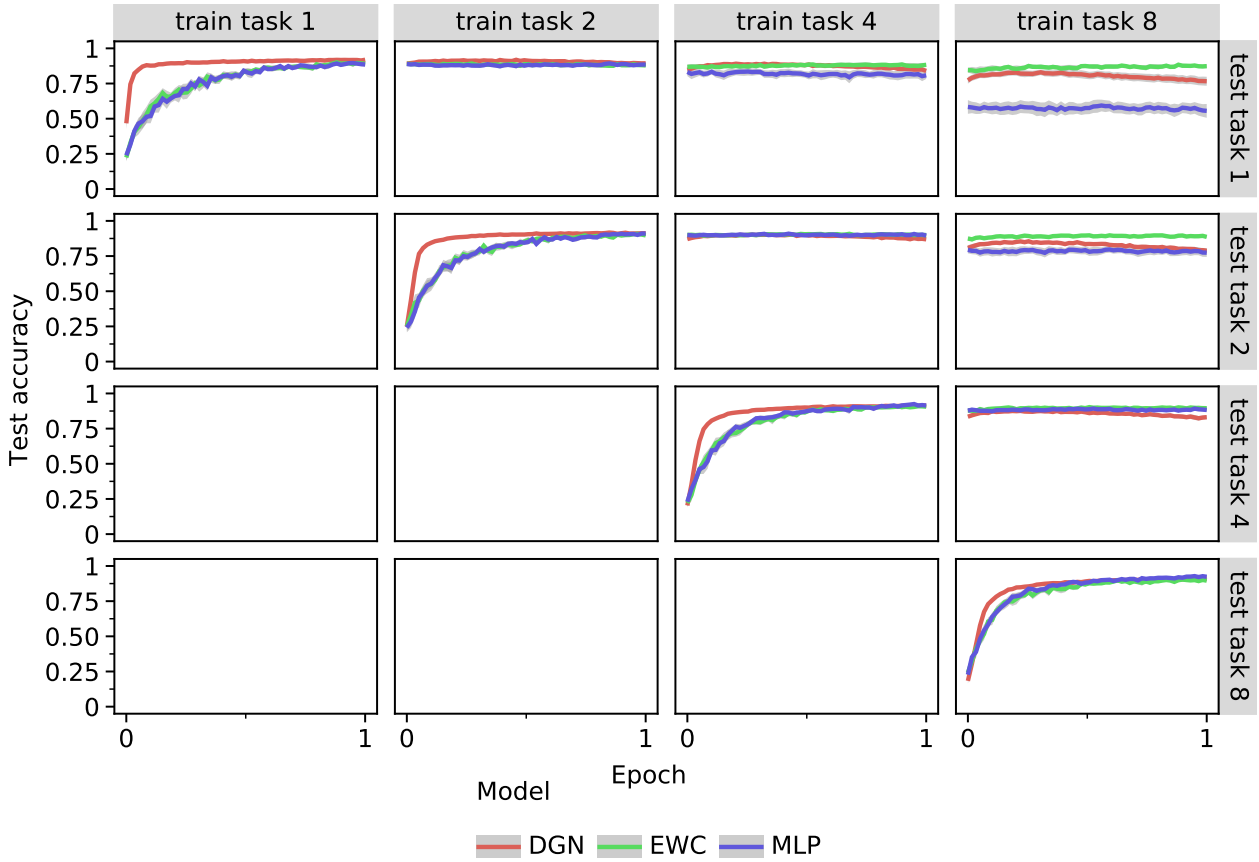

Figure S1: Retention results for permuted MNIST. Models are trained sequentially on 8 tasks (rows) and evaluated on all previously encountered tasks (columns). For example, the top row indicates performance on task 1 after being trained sequentially on tasks 1, 2, 4 and 8. Each model trains for one epoch per task. Error bars, indicated by the thickness of the lines, denote 95% confidence levels over 20 random seeds.

Pseudocode

Algorithm S1 DGN for quadratic loss

Input: network architecture: number of layers $K \in \mathbb{N}$,
number of neurons in layer k $\{n_k \in \mathbb{N}\}$,
number of branches per neuron i in layer k $\{B_{k,i} \in \mathbb{N}\}$
Input: weights $\{w^b_{k,ij} \in \mathbb{R}\}$
Input: gating parameters $\{v^b_{k,ij} \in \mathbb{R}\}$, $\{\theta^b_{k,i} \in \mathbb{R}\}$
Input: input $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{R}^n$
Input: target $r^* \in \mathbb{R}$
Input: learning rate $\eta \in (0, 1)$
Input: update $\in \{\text{TRUE}, \text{FALSE}\}$ (enables learning)
Output: Target prediction $\hat{r} = r_{K,1}$ (output of neuron in last layer K)
$r_{0,0} \leftarrow 1$; $n_0 \leftarrow n$; $r_{0,i} = x_i$ for $i \in \{1, ..., n\}$
for $k \in \{1, \ldots, K\}$ **do** {over layers}
  $r_{k,0} \leftarrow 1$ {bias}
  for $i \in \{1, \ldots, n_k\}$ **do** {over neurons}
    for $b \in \{1, \ldots, B_{k,i}\}$ **do** {over branches}
      $g^b_{k,i} \leftarrow \Theta(\sum_{j=0}^{n_{k-1}} v^b_{k,ij} x_j - \theta^b_{k,i})$
    $r_{k,i} \leftarrow \sum_{b=1}^{B_{k,i}} g^b_{k,i} \sum_{j=0}^{n_{k-1}} w^b_{k,ij} r_{k-1,j}$
    if update **then**
      for $b \in \{1, \ldots, B_{k,i}\}$ **do** {over branches}
        if $g^b_{k,i} > 0$ **then**
          for $j \in \{1, ..., n_{k-1}\}$ **do** {over neurons in previous layer}
            $w^b_{k,ij} \leftarrow w^b_{k,ij} - \eta\,(r_{k,i} - r^*)\, w^b_{k,ij} r_{k-1,j}$
return $r_{K,1}$

where $\Theta(\cdot)$ is the Heaviside step function ($\Theta(z) = 1$ for $z > 0$ and $\Theta(z) = 0$ otherwise).

Algorithm S2 DGN for Bernoulli data

Input: network architecture: number of layers $K \in \mathbb{N}$,
number of neurons in layer k $\{n_k \in \mathbb{N}\}$,
number of branches per neuron i in layer k $\{B_{k,i} \in \mathbb{N}\}$
Input: weights $\{w^b_{k,ij} \in \mathbb{R}\}$
Input: gating parameters $\{v^b_{k,ij} \in \mathbb{R}\}$, $\{\theta^b_{k,i} \in \mathbb{R}\}$
Input: precision $\epsilon \in (0, 0.5)$
Input: input $\mathbf{x} = (x_1, ..., x_n) \in \mathbb{R}^n$
Input: target $r^* \in \{0, 1\}$
Input: learning rate $\eta \in (0, 1)$
Input: update $\in \{\text{TRUE}, \text{FALSE}\}$ (enables learning)
Output: Target prediction $\hat{r} = r_{K,1}$ (output of neuron in last layer K)
$r_{0,0} \leftarrow \sigma(1)$; $n_0 \leftarrow n$; $r_{0,i} = \text{CLIP}^{1-\epsilon}_{\epsilon}(\sigma(x_i))$ for $i \in \{1, ..., n\}$
for $k \in \{1, \ldots, K\}$ **do** {over layers}
  $r_{k,0} \leftarrow \sigma(1)$ {bias}
  for $j \in \{1, \ldots, n_{k-1}\}$ **do** {over neurons in layer below}
    $h_{k-1,j} \leftarrow \sigma^{-1}(r_{k-1,j})$
  for $i \in \{1, \ldots, n_k\}$ **do** {over neurons}
    for $b \in \{1, \ldots, B_{k,i}\}$ **do** {over branches}
      $g^b_{k,i} \leftarrow \Theta(\sum_{j=0}^{n_{k-1}} v^b_{k,ij} x_j - \theta^b_{k,i})$
    $h_{k,i} \leftarrow \sum_{b=1}^{B_{k,i}} g^b_{k,i} \sum_{j=0}^{n_{k-1}} w^b_{k,ij} h_{k-1,j}$
    $r_{k,i} \leftarrow \text{CLIP}^{1-\epsilon}_{\epsilon} \sigma(h_{k,i})$
    if update **then**
      for $b \in \{1, \ldots, B_{k,i}\}$ **do** {over branches}
        if $|r^* - \sigma(h_{k,i})| > \varepsilon$ **then**
          for $j \in \{1, ..., n_{k-1}\}$ **do** {over neurons in previous layer}
            $w^b_{k,ij} \leftarrow w^b_{k,ij} - \eta(r_{k,i} - r^*) h_{k-1,j}$
return $r_{K,1}$

where $\text{CLIP}^b_a(\cdot)$ clips values between a and b,

$$\text{CLIP}^b_a(y) \equiv \begin{cases} a & y < a \\ y & a < y < b \\ b & b \leq y \end{cases} . \tag{12}$$

$\sigma(\cdot)$ is the sigmoid function, $\sigma(z) = e^z/(1 + e^z)$, and $\sigma^{-1}(\cdot)$, its inverse, is given by $\sigma^{-1}(y) = \log(y/(1 - y))$.