PNAS
www.pnas.org

Supplementary Information for

Fundamental bounds on learning performance in neural circuits

Dhruva V. Raman, Adriana Perez Rotondo, Timothy O'Leary

Timothy O'Leary
E-mail: tso24@cam.ac.uk
Dhruva V. Raman
dvr23@cam.ac.uk

This PDF file includes:

Supplementary text
Figs. S1 to S3
References for SI reference citations

www.pnas.org/cgi/doi/10.1073/pnas.1813416116

Supporting Information Text

Supplementary methods

A. Network architectures. The simplest considered network (used only in the first part of the section: *Network expansions that increase learning performance*, and for Figure 6A of the main paper) is linear. Given an input $u \in \mathbb{R}^i$, this gives an output of the form

$$y(u) = Wu \in \mathbb{R}^o,$$

where $W \in \mathbb{R}^{oi}$ is a matrix of synaptic weights.

More commonly we consider networks with nonlinearities and hidden layers. In the main text, we refer to these as nonlinear, feedforward networks. Each neuron in these networks passes inputs through a sigmoidal nonlinearity $\sigma : \mathbb{R} \to \mathbb{R}$ of the form

$$\sigma(x) = \frac{1}{1 + \exp(-x)}.$$

We use the notation $\underline{\sigma} : \mathbb{R}^m \to \mathbb{R}^m$ to represent the elementwise application of σ to a vector of arbitrary length $m \in \mathbb{N}$. Let us denote neural activity at the k^{th} layer via a vector h^k. If $k > 1$, then

$$h^k = \underline{\sigma}(W^k \tilde{h}^{k-1}),$$

where W^k is a matrix of synaptic weights and $\tilde{h}^{k-1}$ is the concatenation of the vector h^{k-1} with the scalar -1. This scalar is known as a bias neuron, and allows for nonzero h^k even when each component of h^{k-1} is zero. Meanwhile for $k = 1$, we have

$$h^1 = \underline{\sigma}(W^1 u),$$

where $u \in \mathbb{R}^i$ is a vector of inputs to the network. Let F be the final hidden layer of the network. Then network output $y \in \mathbb{R}^o$ is taken as

$$y = \underline{\sigma}(W^{F+1} \tilde{h}^F),$$

which will subsequently be denoted $y(\mathbf{w}, u)$. Here $\mathbf{w}$ is a vector that concatenates all entries of each weight matrix in the network.

B. Details of the Learning Tasks. All training tasks used in simulation employ the student-teacher framework. The basic setup is as follows: We first make a teacher network, which has the same basic network architecture as the students (as described in section A). We then initialise the teacher weights at fixed values, which are generated as follows (unless otherwise specified): At the k^{th} layer of the teacher network, weights are distributed uniformly on the interval $[-a_k, a_k]$. Here a_k is set so that the standard deviation of weights on the layer is $\frac{4}{\sqrt{i}}$, where i is the number of inputs to the layer. This scaling with i ensures that the magnitude of hidden layer outputs does not increase with the size of hidden layer (1).

We then specify a set $\mathcal{U}$ consisting of 1000 input vectors. We generate each vector $u \in \mathcal{U}$ componentwise, randomly drawing each component u_i from the distribution $u_i \sim \mathcal{N}(0, 1)$. Note that in Figure 2 of the main paper, this acts as a test set, which is not used when training the network (details in the next section).

The output vector of the teacher, given an input vector u, is denoted $y^*(u)$. The task of the student network, with weights $\mathbf{w}$, is then to match its output to $y^*(u)$, for all $u \in \mathcal{U}$. Therefore the task error is

$$F[\mathbf{w}] = \sum_{u \in \mathcal{U}} \|y^*(u) - y(\mathbf{w}, u)\|_2^2, \qquad [S.1]$$

where $y(\mathbf{w}, u)$ denotes the output of the student given input u and weights $\mathbf{w}$.

For the linear networks studied at the beginning of the section: *Network expansions that increase learning performance*, the dimensionality of the input and output vectors differ between the teacher and the students. However, fixed matrix transformations lift the teacher inputs/outputs into the appropriate dimensionality (see Eq. (S.13)). For all other networks, the number of network inputs/outputs is shared between the students and the teacher. Teacher and students also have the same number of hidden layers. At the i^{th} hidden layer, each student has at least as many neurons as the teacher. This ensures that the teacher network forms a subset of each student network, and therefore that each student network is theoretically capable of exactly recreating the input-output mapping $y^*(u)$ of the teacher.

C. Network Training. The theoretical analysis in the main paper, as well as the simulations of Figure 2, account for online learning. Online learning typically considers sampling data points from an infinite distribution of data. However in cases where we needed to compute a true gradient (specifically Figures 4 and 6 of the main paper), we needed to define finite distributions to numerically evaluate the true gradient. Having the true gradient allows us to precisely specify values of task-relevant and task-irrelevant components of plasticity.

We emphasize (as we emphasized in the main text) that this differs from treating the finite set as a sample - or batch - from an infinite distribution, which would incur generalization issues because any finite sample will be necessarily biased. The relationship between the results of our paper and the latter scenario are described in the SI section *Regularization and generalization error*.

We first describe the training of the numerical simulations in Figures 4 and 6 of the main paper, where learning is conducted on the finite input set $\mathcal{U}$ described in the previous section. At each learning cycle, we apply the weight update

$$\mathbf{w}_{t+T} = \mathbf{w}_t - T\bar{\gamma}_1 \nabla \hat{F}[\mathbf{w}_t] + T\bar{\gamma}_2 \hat{\mathbf{n}}_2^t + \sqrt{NT}\bar{\gamma}_3 \hat{\mathbf{n}}_3^t. \quad [S.2]$$

The hyper-parameters are $\{\bar{\gamma}_i\}_{i=1}^3$ and T, which collectively specify the quality of the learning rule, the feedback delay, and the intrinsic, per-synapse noise (see main text). N is the number of synapses in the network. We calculate the gradient $\nabla F[\mathbf{w}_t]$ by taking the gradient of Eq. (S.1) using backpropagation. The reason we use the notation $\bar{\gamma}$ here, as opposed to γ in the main text, will be made clear later in the section. The normalised vectors $\hat{\mathbf{n}}_2^t$ and $\hat{\mathbf{n}}_3^t$ represent sources of task-irrelevant plasticity. The dynamics of the unnormalised vector $\mathbf{n}_2$ satisfies $\mathbf{n}_2^{t+1} = \sqrt{0.1}\hat{\mathbf{n}}_2^t + \sqrt{0.9}\hat{\nu}$, where ν is a Gaussian random variable, normalised such that $\|\nu\|_2 = 1$. This endows the task-irrelevant plasticity vector $\hat{\mathbf{n}}_2$ with some amount of temporal autocorrelation. On the other hand $\hat{\mathbf{n}}_3$, which models intrinsic synaptic noise, is a Gaussian random variable, normalised so that $\|\hat{\mathbf{n}}_3\|_2 = 1$.

Note that $\mathbb{E}[\langle \hat{\mathbf{n}}_i, \nabla \hat{F}[\mathbf{w}_t]]\rangle = 0$. However, on a specific learning cycle, the actual quantity $\langle \hat{\mathbf{n}}_i, \nabla \hat{F}[\mathbf{w}_t]]\rangle$ may be nonzero. Therefore the true values of γ_i, as defined in equation (4) of the main text, will fluctuate between learning cycles, around a mean $\bar{\gamma}_i$. For instance, if $\hat{\mathbf{n}}_2$ anticorrelates with the gradient by chance on a particular learning cycle, then $\frac{\bar{\gamma}_1}{\bar{\gamma}_2} < \frac{\gamma_1}{\gamma_2}$. The opposite is true if the correlation is positive.

Network training is different in Figures 2C and 2D of the main paper: it is conducted online from an infinite distribution. At each learning cycle, we randomly draw a single input vector u of Gaussian components, i.e. $u_i \sim \mathcal{N}(0,1)$. We replace the term $\nabla \hat{F}[\mathbf{w}_t]$ in the weight update equation Eq. (S.2) with $\nabla \hat{F}[\mathbf{w}, u]$, the (normalised) stochastic gradient, where $F[\mathbf{w}, u] := \|y^*(u) - y(\mathbf{w}, u)\|_2^2$. The overall error $F[\mathbf{w}]$ is then the expected error on the next input, i.e.

$$F[\mathbf{w}] = \int_\Omega F[\mathbf{w}, u]\mathbb{P}[u]\ du,$$

where $\mathbb{P}[u]$ is the componentwise Gaussian probability density function from which u is drawn, with support Ω. We cannot exactly calculate $F[\mathbf{w}]$ in this setting, as we have to integrate over a distribution. However, the error function described in the previous section: *Details of the Learning Tasks*, which is constructed from 1000 inputs, is an un-normalized estimate of $F[\mathbf{w}]$. So we use this as the error function depicted on the y-axis of these Figures. Note we also cannot compute the gradient $\nabla F[\mathbf{w}]$ for the same reason. This prevents us from setting explicitly the magnitudes of task-(ir)relevant plasticity (i.e. γ_1 and γ_2 values) during learning in the online setting.

D. Student weight initialization. The initialization of the random fixed weights of the teacher network was described in Section B. We pre-initialize the weights of the smallest student network in exactly the same way. We then train this student for 500 learning cycles exactly as described in the rest of the methods, but using a different, 'fake' teacher network. 'Fake' teachers are generated probabilistically in exactly the same way as real teachers. The student weights after this 'fake' training are taken to be the initial student weights. This ensures that the statistical distribution of the initial student weights reflects that of the student weights after training.

We then set the initial weights of the remaining (larger) students in such a way that they exhibit the exact same input-output properties as the smallest student. This ensures that $F[\mathbf{w}(0)]$ is identical for all sizes of network, necessary for a fair test. This is described mathematically in detail in SI sections *Learning in a linear network* and *Learning in a nonlinear feedforward network* below.

Supplementary notes on mathematical analyses

Necessary Conditions on Learning. In this section, we specify some necessary requirements on any learning rule, and provide insight into factors that make a task 'easier' to learn. We consider a situation in which a neural network has to learn a task, by changing its synaptic strengths (represented in a vector $\mathbf{w}$). The level of training is represented by some task-error function $F[\mathbf{w}]$, which decreases with increasing levels of performance. In order to learn, the network receives some form of sensory feedback on task error, which informs changes at each synapse through some unknown learning rule.

We will first ask what it takes for a network to learn, given limited rates of sensory feedback on task error, over some time interval $[0, T]$. Clearly, learning is equivalent to a decrease in task error, which corresponds to

$$F[\mathbf{w}(T)] - F[\mathbf{w}(0)] < 0.$$

The degree of learning is determined by how much smaller than zero this quantity is. Suppose our task error is continuously differentiable in $\mathbf{w}$. Then, by the Fundamental Theorem of Calculus,

$$\begin{aligned} F[\mathbf{w}(T)] - F[\mathbf{w}(0)] &= \int_0^T \langle \nabla F[\mathbf{w}(t)], \dot{\mathbf{w}}(t) \rangle \, dt \\ &= T \, \mathbb{E}_t[\langle \nabla F[\mathbf{w}(t)], \dot{\mathbf{w}}(t) \rangle], \end{aligned} \tag{S.3}$$

where expectation is taken across a uniform distribution of timepoints on the interval $[0, T]$, and $\dot{\mathbf{w}}(t)$ denotes the time-derivative of $\mathbf{w}$. Let us pick (uniformly) a random synaptic weight w_i, at a (uniformly) random timepoint $t \in [0, T]$. If learning has occurred, then the above equations imply

$$\mathbb{E}\big[\big(\nabla F[\mathbf{w}(t)]_i\big)\big(\dot{w}_i(t)\big)\big] < 0.$$

In other words, at time t, we expect the average synapse to be changing in a direction that anticorrelates with $\nabla F[\mathbf{w}(t)]_i$, the i^{th} component of the gradient of task performance at time t. Therefore, any learning rule must possess (on average) information on $\nabla F[\mathbf{w}(t)]$ at time t. However, any learning rule will be using 'old' information at time t. Information on task error may only be supplied intermittently, and there must exist some biochemically induced delay between sensory acquisition of information on task error, and its integration into plasticity rules at each synapse. How well a learning rule performs therefore depends on how relevant 'old' information is.

We can obtain intuition on the previous assertion. Suppose the plasticity direction $\dot{\mathbf{w}}(t)$ was determined using 'old' information from time 0. Suppose that the learning rule calculated an estimate $\nabla \tilde{F}[\mathbf{w}(0)]$ of $\nabla F[\mathbf{w}(0)]$. If the gradient was constant over the time interval $[0, t]$, then we would have $\nabla F[\mathbf{w}(0)] = \nabla F[\mathbf{w}(t)]$, and the delay from time 0 to time t would have no effect on the quality of the estimate. Conversely if the gradient changed very fast over this period, then the delay would greatly decrease the quality of the estimate. What influences how fast the gradient's direction changes? Geometrically, the rate of change of the gradient is the curvature, or second derivative, of $F[\mathbf{w}]$. Thus the geometry of $F[\mathbf{w}]$ is one factor. Another is the overall speed of synaptic change: if the network moves faster through weight space, then the gradient direction will also change faster. Task irrelevant plasticity from synaptic processes unconnected with learning also contributes to this speed, and therefore hinders learning in general.

One of the results in the paper is a demonstration of how increasing network size can 'flatten out' an error function $F[\mathbf{w}]$. This flattening makes any learning rule with unavoidable amounts of both delay between information acquisition and synaptic plasticity, and task-irrelevant plasticity, learn faster.

Learning rate and local task difficulty. In order to quantify learning rate over the time interval $[0, T]$, we take k such that

$$F[\mathbf{w}(T)] = [1 - kT]F[\mathbf{w}(0)].$$

Meanwhile we represent overall synaptic change over the time interval $[0, T]$, normalised by T, as

$$\dot{\boldsymbol{\omega}}_T = \frac{\mathbf{w}(T) - \mathbf{w}(0)}{T}.$$

We perform a second order Taylor expansion of the above equation to get:

$$-kTF[\mathbf{w}(0)] = F[\mathbf{w}(T)] - F[\mathbf{w}(0)] \tag{S.4a}$$

$$= T\langle \nabla F[\mathbf{w}(0)], \dot{\boldsymbol{\omega}}_T \rangle + \frac{1}{2} T^2 \langle \dot{\boldsymbol{\omega}}_T, \nabla^2 F[\mathbf{w}(0)] \dot{\boldsymbol{\omega}}_T \rangle + \mathcal{O}(T^3). \tag{S.4b}$$

We can rearrange this equation to make k the subject. This gives

$$k = \frac{-\|\nabla F[\mathbf{w}(0)]\|_2}{F[\mathbf{w}(0)]} \left[\left\langle \dot{\boldsymbol{\omega}}_T, \nabla \hat{F}[\mathbf{w}(0)] \right\rangle + \mathbf{G}_F[\dot{\hat{\boldsymbol{\omega}}}_T] \|\dot{\boldsymbol{\omega}}_T\|_2^2 T \right] + \mathcal{O}(T^2), \tag{S.5a}$$

where

$$\mathbf{G}_F[\dot{\hat{\boldsymbol{\omega}}}_T] := \frac{1}{2\|\nabla F[\mathbf{w}(0)]\|_2} \left\langle \dot{\hat{\boldsymbol{\omega}}}_T, \nabla^2 F[\mathbf{w}(0)] \dot{\hat{\boldsymbol{\omega}}}_T \right\rangle. \tag{S.5b}$$

Decomposition of local task difficulty . In the main text $\mathbf{n}_2$ and $\mathbf{n}_3$ correspond to different sources of task-irrelevant synaptic changes (see equation 4). We treat them as random variables arising from some unknown probability distribution. Since they are generated by task-independent processes, they are uncorrelated with derivatives of the task error. Therefore

$$\mathbb{E}\left[\left\langle \nabla F[\mathbf{w}(0)], \nabla^2 F[\mathbf{w}(0)]\mathbf{n}_i \right\rangle\right] = 0 \quad \text{for } i \in \{2,3\}. \tag{S.6a}$$

We also assume that $\mathbb{E}[n_{2,i}n_{3,i}] = 0$, for any component i, i.e. the two sources of plasticity are uncorrelated with each other. This follows from the fact that $\mathbf{n}_3$ is the result of a white-noise process evolving at each synapse, and is thus uncorrelated with any random variable that is not itself derived from the same white-noise process. A consequence of this assumption is that

$$\mathbb{E}\left[\left\langle \mathbf{n}_2, \nabla^2 F[\mathbf{w}(0)]\mathbf{n}_3 \right\rangle\right] = 0. \tag{S.6b}$$

We now substitute our expanded expression (see equation 5 of the main text)

$$\dot{\boldsymbol{\omega}}_T = -\gamma_1 \nabla \hat{F}[\mathbf{w}(0)] + \gamma_2 \hat{\mathbf{n}}_2 + \gamma_3 \sqrt{\frac{N}{T}} \hat{\mathbf{n}}_3. \tag{S.7}$$

for the interpolated synaptic velocity into our formula for the local task difficulty $\mathbf{G}_F[\dot{\hat{\boldsymbol{\omega}}}_T]$. We can use equations Eq. (S.6) to simplify the consequent expression, and thereby get

$$\mathbb{E}\left[\mathbf{G}_F[\dot{\hat{\boldsymbol{\omega}}}_T]\right] \|\dot{\boldsymbol{\omega}}_T\|_2^2 = \gamma_1^2 \mathbf{G}_F^1[\mathbf{w}(0)] + \gamma_2^2 \mathbf{G}_F^2[\dot{\hat{\boldsymbol{\omega}}}_T] + \gamma_3^2 \frac{N}{T} \mathbf{G}_F^3[\dot{\hat{\boldsymbol{\omega}}}_T], \tag{S.8}$$

where

$$\mathbf{G}_F^1[\mathbf{w}(0)] = \frac{1}{2\|\nabla F[\mathbf{w}(0)]\|_2} \left\langle \nabla \hat{F}[\mathbf{w}(0)], \nabla^2 F[\mathbf{w}(0)] \nabla \hat{F}[\mathbf{w}(0)] \right\rangle \tag{S.9a}$$

$$\mathbf{G}_F^2[\dot{\hat{\boldsymbol{\omega}}}_T] = \frac{1}{2\|\nabla F[\mathbf{w}(0)]\|_2} \left\langle \hat{\mathbf{n}}_2, \nabla^2 F[\mathbf{w}(0)] \hat{\mathbf{n}}_2 \right\rangle \tag{S.9b}$$

$$\mathbf{G}_F^3[\dot{\hat{\boldsymbol{\omega}}}_T] = \frac{1}{2\|\nabla F[\mathbf{w}(0)]\|_2} \left\langle \hat{\mathbf{n}}_3, \nabla^2 F[\mathbf{w}(0)] \hat{\mathbf{n}}_3 \right\rangle \tag{S.9c}$$

The independence of $\mathbf{n}_2$ and $\mathbf{n}_3$ from $\nabla^2 F[\mathbf{w}]$ allows us to further simplify the expressions for $\mathbf{G}_F^2$ and $\mathbf{G}_F^3$. Specifically, we can write $\mathbf{n}_i$ as

$$\mathbf{n}_i = \sum_{j=1}^{N} c_j \mathbf{v}_j,$$

where $\mathbf{v}_j$ denotes the j^{th} eigenvector of $\nabla^2 F[\mathbf{w}]$. Independence of $\mathbf{n}_i$ from $\nabla^2 F[\mathbf{w}]$ implies that $\mathbb{E}[c_j] = \mathbb{E}[c_k]$ for any $j, k \in \{1, \ldots, N\}$. This gives

$$\mathbb{E}\left\langle \hat{\mathbf{n}}_i, \nabla^2 F[\mathbf{w}(0)] \hat{\mathbf{n}}_i \right\rangle = \frac{Tr(\nabla^2 F[\mathbf{w}(0)])}{N}, \tag{S.10}$$

the mean of the eigenvalues of $\nabla^2 F[\mathbf{w}(0)]$. With this, equation Eq. (S.8) becomes

$$\mathbb{E}\left[\mathbf{G}_F[\dot{\hat{\boldsymbol{\omega}}}_T]\right] \|\dot{\boldsymbol{\omega}}_T\|_2^2 = \gamma_1^2 \mathbf{G}_F^1[\mathbf{w}(0)] + \frac{Tr(\nabla^2 F[\mathbf{w}(0)])}{2\|\nabla F[\mathbf{w}(0)]\|_2} \left[\frac{\gamma_2^2}{N} + \frac{\gamma_3^2}{T}\right]. \tag{S.11}$$

Task-relevant Plasticity. Recall that $\dot{\boldsymbol{\omega}}_T$ represents the vector of synaptic weight changes over a time interval $[0, T]$. A learning rule must determine both the speed and direction of synaptic plasticity, and thus $\dot{\boldsymbol{\omega}}_T$, over this time interval, using any task-related information it possesses. Suppose that the speed $\|\dot{\boldsymbol{\omega}}_T\|_2$ of this change was fixed at an arbitrary level. It remains for the learning rule to determine a direction $\dot{\hat{\boldsymbol{\omega}}}_T$ of synaptic plasticity over the time interval. If the objective of the learning rule was to minimise $F[\mathbf{w}(0) + \dot{\boldsymbol{\omega}}_T]$, the task error at time t, which direction should it pick?

First note that we can use a Taylor expansion to express the change in task error over $[0, T]$ as

$$F[\mathbf{w}(0) + T\dot{\boldsymbol{\omega}}_T] - F[\mathbf{w}(0)] = T\|\dot{\boldsymbol{\omega}}_T\|_2 \langle \dot{\hat{\boldsymbol{\omega}}}_T, \nabla F[\mathbf{w}(0)] \rangle + T^2 \|\dot{\boldsymbol{\omega}}_T\|_2^2 \langle \dot{\hat{\boldsymbol{\omega}}}_T, \nabla^2 F[\mathbf{w}(0)] \dot{\hat{\boldsymbol{\omega}}}_T \rangle + \mathcal{O}[T^3 \|\dot{\boldsymbol{\omega}}_T\|_2^3]. \tag{S.12}$$

Maximising the degree of learning involves minimising this quantity.

In the main text, we consider learning rules that have (at best) information on the gradient $\nabla F[\mathbf{w}(0)]$, but not on higher order derivatives $\nabla^{(n)} F[\mathbf{w}(0)]$. These are known as first-order learning rules. Given such information, a learning rule can only attempt to minimise the first term in Eq. (S.12). Hence, the optimal direction of plasticity would be the solution of

$$\begin{aligned} \dot{\hat{\boldsymbol{\omega}}}_T^{*,1} &= \min_{\hat{x}} T\|\dot{\boldsymbol{\omega}}_T\|_2 \langle \hat{x}, \nabla F[\mathbf{w}(0)] \rangle : \qquad \|\hat{x}\|_2 = 1 \\ &= -\nabla \hat{F}[\mathbf{w}(0)]. \end{aligned}$$

Indeed we define $-\hat{F}[\mathbf{w}(0)]$ (i.e. $\dot{\hat{\boldsymbol{\omega}}}_T^{*,1}$) as the direction of task-relevant plasticity in the main text. The expected correlation of additional, task-irrelevant plasticity with the Hessian $\nabla^2 F[\mathbf{w}(0)]$ is taken as zero, as must be true of any learning rule without access to information on the Hessian.

Learning rules that additionally have information on $\nabla^2 F[\mathbf{w}(0)]$ are known as second order learning rules. Given the additional information, $-\nabla\hat{F}[\mathbf{w}(0)]$ is no longer the best direction of plasticity. Instead, the optimal direction would minimise the first two terms of Eq. (S.12), and so would be

$$\dot{\hat{\boldsymbol{\omega}}}_T^{*,2} = \min_{\hat{x}} T\|\dot{\boldsymbol{\omega}}_T\|_2\langle\hat{x}, \nabla F[\mathbf{w}(0)]\rangle + T^2\|\dot{\boldsymbol{\omega}}_T\|_2^2\langle\hat{x}, \nabla^2 F[\mathbf{w}(0)]\hat{x}\rangle : \qquad \|\hat{x}\|_2 = 1.$$

Whereas $\dot{\hat{\boldsymbol{\omega}}}_T^{*,1}$ was a steepest descent direction of the task error, $\dot{\hat{\boldsymbol{\omega}}}_T^{*,2}$ looks for directions that balance immediate descent with both downward curvature of the task error, and the time T until direction can be updated. Intuitively, a descent direction that is downwardly curved is likely to remain a descent direction as the weights change along it.

The solution to this minimisation has a complicated analytic form that is dependent upon both T and $\|\dot{\boldsymbol{\omega}}_T\|_2$. In order to incorporate second order learning rules in the main text, we would have to change the direction of task-relevant plasticity from $\dot{\hat{\boldsymbol{\omega}}}_T^{*,1}$ to $\dot{\hat{\boldsymbol{\omega}}}_T^{*,2}$. How would using $\dot{\hat{\boldsymbol{\omega}}}_T^{*,2}$ as the direction of task-relevant plasticity change the results of the main text? All workings in the paper would have to replace the original formula (see equation Eq. (S.7)) for $\dot{\boldsymbol{\omega}}_T$ with

$$\dot{\boldsymbol{\omega}}_T = -\gamma_1\dot{\hat{\boldsymbol{\omega}}}_T^{*,2} + \gamma_2\hat{\mathbf{n}}_2 + \gamma_3\sqrt{\frac{N}{T}}.$$

This is tractable, but technically cumbersome. Of particular note is that

$$\langle\dot{\hat{\boldsymbol{\omega}}}_T^{*,2}, \nabla^2 F[\mathbf{w}(0)]\dot{\hat{\boldsymbol{\omega}}}_T^{*,2}\rangle \leq \langle\dot{\hat{\boldsymbol{\omega}}}_T^{*,1}, \nabla^2 F[\mathbf{w}(0)]\dot{\hat{\boldsymbol{\omega}}}_T^{*,1}\rangle.$$

Consequently, the formula for Eq. (S.9a), the component of local task difficulty attributable to task-relevant plasticity, would decrease, as we would get

$$\mathbf{G}_F^1[\mathbf{w}(0)] = \frac{1}{2\|\nabla F[\mathbf{w}(0)]\|_2}\left\langle\dot{\hat{\boldsymbol{\omega}}}_T^{*,2}, \nabla^2 F[\mathbf{w}(0)]\dot{\hat{\boldsymbol{\omega}}}_T^{*,2}\right\rangle.$$

Qualitatively, the results of the paper would not change in that the relationship between task-irrelevant plasticity and network size would be preserved. Task-irrelevant plasticity would still be uncorrelated with the derivatives of task error. So we would arrive at an identical equation Eq. (S.11), except for the changed formula of $\mathbf{G}_F^1[\mathbf{w}(0)]$.

Note that we could analogously consider n^{th} order learning rules, for arbitrary n, by taking the direction of task-relevant plasticity as $\dot{\hat{\boldsymbol{\omega}}}_T^{*,n}$, where this direction minimises the first n terms of the Taylor expansion Eq. (S.12). A learning rule with access to derivatives of all orders n, given an analytic task-error function, is equivalent to a learning rule that knows *a priori* the optimal weight configuration of the network, and induces synaptic plasticity in a direction pointing directly towards this optimal configuration. We provide this observation for intuition only, and do not suggest that such a learning rule would realistically exist in a biological context.

Learning in a Linear Network. We consider the simple case of a linear network undertaking supervised learning with a quadratic error function. The general (nonlinear) case is considered subsequently. Inputs u are drawn from some space $\mathcal{U} \subseteq \mathbb{R}^i$ with probability $\mathbb{P}(u)$. For each input u, there is an optimal output (i.e. label) $y^*(u) \in \mathbb{R}^o$.

The network linearly transforms any input into an output $y = Wu$, for a matrix $W \in \mathbb{R}^{oi}$ of synaptic weights. Note the cosmetic difference between the matrix W, and previous representations of the synaptic weights, which took the form of a vector $\mathbf{w} \in \mathbb{R}^N$. These representations are interchangeable, as we can reshape W into a vector $\mathbf{w}$, and take $N = oi$. Indeed, we will switch between the matrix and vector representations of the weights in the ensuing discussion. The input-dependent error of the network, for any particular input $u \in \mathcal{U}$, is taken as

$$F(W, u) = \|y^*(u) - Wu\|_2^2.$$

Meanwhile, the overall network error integrates input-dependent error over the probability distribution $\mathbb{P}(u)$ of inputs, i.e.

$$F[W] = \int_{u\in\mathcal{U}} F[W, u] \;\; \mathbb{P}(u)\; du.$$

Even if $y^*(u)$ were known for each $u \in \mathcal{U}$, calculation of $F[W]$ would require passage of every input $u \in \mathcal{U}$ through the network. In a biologically realistic setting, $y^*(u)$ would not be known, and both $F[W]$ and its gradient would have to be estimated, using a subset of inputs. So no learning rule minimising the error $F[W]$ could have uncorrupted access to $\nabla F[W]$.

Now let us lift the learning problem into a higher dimensional setting. We will consider a matrix W' with $c_1 i$ inputs and $c_2 o$ outputs, for some constants c_1, $c_2 > 1$. We will define the total number of weights as $\tilde{N} = c_1 i c_2 o$. We take the transformation $u' = Bu \in \mathbb{R}^{c_1 i}$, where $B \in \mathbb{R}^{c_1 i \times i}$ is an arbitrary semi-orthogonal matrix. (i.e. it satisfies $B^T B = \mathbb{I}_i$). Geometrically, B

therefore represents the composition of a projection into the higher dimensional space $\mathbb{R}^{c_1 i}$ with a rotation. Note that this is an invertible mapping: if $u' = Bu$ then $B^T u' = u$. Similarly, we can take $y'^*(u') = Dy^*(u) \in \mathbb{R}^{c_2 o}$, where $D^T D = \mathbb{I}_o$.

The expanded neural network with weights $W' \in \mathbb{R}^{c_1 o \times c_2 i}$ has to learn the same mapping as the original, but with respect to the higher dimensional inputs. So the network receives inputs $u' \in B\mathcal{U}$, and transforms them to outputs $y' = W'u'$, with input-dependent error

$$F'[W', u'] = \|y'^*(u') - W'u'\|_2^2. \tag{S.13a}$$
$$= \|Dy^*(u) - W'Bu\|_2^2 \ \text{ for some } u : u' = Bu \tag{S.13b}$$
$$= \|y^*(u) - D^T W'Bu\|_2^2 \ \text{ by semi-orthogonality of } D. \tag{S.13c}$$

Let us randomly draw a state $W' \in \mathbb{R}^{c_2 o \times c_1 i}$ of the bigger network. For instance, each element of W' can be drawn from a Gaussian distribution. Regardless of W', equations Eq. (S.13) show us that $W = D^T W' B \in \mathbb{R}^{oi}$ defines a state of the original network satisfying

$$F'[W'] = F[W].$$

Differentiating, we get

$$\begin{aligned}
\nabla F'[W', u'] &= 2Bu(u^T B^T W'^T D - y^*(u)^T)D^T \\
\|\nabla F'[W', u']\|_F^2 &= 4\|u\|_2^2 Tr\Big[\big(u^T B^T W'^T D - y^*(u)^T\big)\big(u^T B^T W'^T D - y^*(u)^T\big)^T\Big] \\
&= 4\|u\|_2^2 Tr\Big[\big(u^T W^T - y^*(u)^T\big)\big(u^T W^T - y^*(u)^T\big)^T\Big] \\
&= \|\nabla F[W, u]\|_F^2,
\end{aligned}$$

From this we see that $\|\nabla F'[W']\|_F = \|\nabla F[W]\|_F$, or equivalently $\|\nabla F'[\mathbf{w}']\|_2 = \|\nabla F[\mathbf{w}]\|_2$.

Meanwhile the quadratic nature of the error function implies $\nabla^2 F'[W']$ is constant. This quantity is a four-tensor, written below according to the Einstein summation convention, and where δ denotes the Kronecker delta-function:

$$\nabla^2 F'[W', u'] = 2u'_g B_d{}^g \delta^{ac} B^b{}_m u'^m.$$

The trace of this matrix can be calculated explicitly as

$$Tr(\nabla^2 F'[W', u']) = 2u'_g B_d{}^g \delta^{ac} B^b{}_m u'^m \delta_{ac} \delta^d{}_b \tag{S.14}$$
$$= 2c_2 o \|u\|_2^2. \tag{S.15}$$

From this we see that $Tr(\nabla^2 F'[W']) = c_2 Tr(\nabla^2 F[W])$.

We hereon consider the synaptic weight matrices W' and W as vectors $\mathbf{w}' \in \mathbb{R}^{\tilde{N}}$ and $\mathbf{w} \in \mathbb{R}^N$. We will also assume that $\nabla \hat{F}'[\mathbf{w}']$ (which is a normalised vector) projects approximately equally onto the different eigenvectors of $\nabla^2 F'[\mathbf{w}']$. The latter is fixed, regardless of $\mathbf{w}'$, whereas the former is a linear function of the (randomly chosen) $\mathbf{w}'$, which justifies the approximation. In this case, Eq. (S.15) implies

$$\nabla \hat{F}'[\mathbf{w}']^T \ \nabla^2 F'[\mathbf{w}'] \ \nabla \hat{F}'[\mathbf{w}'] \approx c_2 \nabla \hat{F}[\mathbf{w}]^T \ \nabla^2 F[\mathbf{w}] \ \nabla \hat{F}[\mathbf{w}]. \tag{S.16}$$

Bringing together equation Eq. (S.16) with the formula Eq. (S.9a) for $\mathbf{G}_F^1$, we see that

$$\mathbf{G'}^1_{F'}[\mathbf{w}'] \approx c_2 \mathbf{G}_F^1[\mathbf{w}]. \tag{S.17a}$$

Similarly

$$\frac{Tr(\nabla^2 F'[\mathbf{w}'])}{2\|\nabla F'[\mathbf{w}']\|_2} \approx c_2 \frac{Tr(\nabla^2 F[\mathbf{w}])}{2\|\nabla F[\mathbf{w}]\|_2}. \tag{S.17b}$$

Collectively, equations Eq. (S.17) imply that

$$\mathbb{E}\big[\mathbf{G'}_{F'}[\dot{\hat{\boldsymbol{\omega}}}'_T]\big]\|\dot{\boldsymbol{\omega}}'_T\|_2^2 \approx c_2 \mathbf{G}_F^1[\mathbf{w}] + c_2 \frac{Tr(\nabla^2 F[\mathbf{w}])}{2\|\nabla F[\mathbf{w}]\|_2}\left[\frac{\gamma_2^2}{\tilde{N}} + \frac{\gamma_3^2}{T}\right].$$

Now recall the learning rate equation, which can be rewritten as

$$k = \frac{-\|\nabla F[\mathbf{w}(0)]\|_2}{F[\mathbf{w}(0)]}\Big[-\gamma_1 + \boldsymbol{\delta}\Big] + \mathcal{O}(T^2).$$
$$\boldsymbol{\delta} = \mathbf{G}_F[\dot{\hat{\boldsymbol{\omega}}}_T]\|\dot{\boldsymbol{\omega}}_T\|_2^2 T,$$

with $\mathcal{O}(T^2) \equiv 0$ for a quadratic error function such as we have. As long as we preserve the ratio $\frac{c_2}{c_1}$, equations Eq. (S.17) allow us to consider N as an independent parameter in the equation for $\boldsymbol{\delta}$. This allows us to optimise steady state error of the network by changing N. To see how, suppose the network has reached steady state error, i.e $\mathbb{E}[k] = 0$. If we decreased $\boldsymbol{\delta}$, then $\mathbb{E}[k]$ would increase, and the network would learn further. Therefore, to derive the optimal N^*, we should minimise the expression for $\boldsymbol{\delta}$ in N. This is equivalent to minimising $\boldsymbol{\delta}$ in c_2, if we keep the ratio $\frac{c_2}{c_1}$ fixed. We differentiate $\boldsymbol{\delta}$ in c_2, and note that N^* is a stationary point, thus satisfying:

$$N^* = \frac{\gamma_2^2}{C\gamma_1^2 + \frac{\gamma_3^2}{T}} \tag{S.18}$$

$$\text{where } C = \frac{\langle \nabla \hat{F}[\mathbf{w}(0)], \nabla^2 F[\mathbf{w}(0)] \nabla \hat{F}[\mathbf{w}(0)] \rangle}{Tr(\nabla^2 F[\mathbf{w}(0)])}. \tag{S.19}$$

Note that C is unknown, and depends on the particular weight configuration of the network. However, we can take the heuristic $C \approx \frac{1}{N^*}$. This heuristic is exact if the gradient $\nabla \hat{F}[\mathbf{w}(0)]$ projects equally onto each of the eigenvalues of the $\nabla^2 \hat{F}[\mathbf{w}(0)]$. This in turn would make the numerator of C equal the mean eigenvalue, i.e. $\frac{Tr(\nabla^2 \hat{F}[\mathbf{w}(0)])}{N^*}$. This heuristic should hold on average over time, given that the Hessian $\nabla^2 \hat{F}[\mathbf{w}(0)]$ is fixed throughout learning due to the quadratic error function, while, the direction $\nabla^2 \hat{F}[\mathbf{w}(0)]$ depends completely on $\mathbf{w}(0)$, which changes over each learning cycle in a direction independent of $\nabla^2 \hat{F}[\mathbf{w}(0)]$.

With this heuristic, we get

$$N^* \approx \frac{T\gamma_2^2}{\gamma_3^2}\left(1 - \frac{\gamma_1^2}{\gamma_2^2}\right). \tag{S.20}$$

Note that $\boldsymbol{\delta}$ is monotonically increasing in $\tilde{N}$, for $\tilde{N} > N^*$. Therefore if $N^* < N$, then the optimal size of the student network is N, its smallest possible value (i.e. any added redundancy hurts steady state error). Otherwise, N^* defines the optimal network size. Critical to our derivation was the ability to take an arbitrary state W' of the larger network, and find a corresponding state W of the nominal network that was linked to W' via a transformation that preserved task error. This will not be the case in subsequent, more general examples.

In summary, we have considered linear networks with quadratic error functions learning a random mapping. We have provided a transformation that increases the size of the linear network, and embeds the random mapping in a higher dimensional space. The existence of this transformation assures us that the higher-dimensional learning problem should have a lower task difficulty. Given a synaptic learning rule with known levels of gradient information, systematic error, and intrinsic noise, we can predict a priori what the optimal dimensionality of the linear network should be, if we want the network to minimise steady state error.

Learning in a nonlinear, feedforward network. We now extend to the case of multilayer, feedforward networks with nonlinearities, before finally considering the general case. We will consider a static nonlinearity $\sigma : \mathbb{R} \to \mathbb{R}$, which can be overloaded to give a function $\underline{\sigma} : \mathbb{R}^m \to \mathbb{R}^m$ representing the elementwise application of σ to a vector of length $\mathbb{R}^m$, for arbitrary $m \in \mathbb{N}$. The k^{th} layer of the network transforms any input $u^{(k-1)}$ from the previous layer into an output of the form

$$y^{(k)} = \underline{\sigma}(W^{(k)} u^{(k-1)}).$$

Again, input-dependent error is taken as the squared deviation between the network output and some optimal output $y^*(u)$. We take W as the concatenation of the weight matrices $W^{(k)}$ at each layer k. For instance, in the case of a two-layer network, we have

$$F(W, u) = \|y^*(u) - \sigma\Big(W^{(2)}\sigma(W^{(1)}u)\Big)\|_2^2.$$

In the linear example, we considered a random state (i.e. weight configuration) of the expanded network, and transported it to an equivalent state of the nominal network via an error-preserving transformation. In the present example this is not possible, as the expanded network can exhibit a fundamentally higher-dimensional set of behaviours. Instead, we will construct an error-preserving transformation ϕ in the opposite direction, from states of the nominal network to states of the larger network. This transformation will essentially add neurons to the network so that gradient information is distributed over a larger number of neurons. Our analysis will quantify the effect of this network expansion on local task difficulty, and hence determine an optimal network size. Technically, it will only hold for network states of the expanded network that reside in the image of the mapping ϕ. However, we subsequently justify why extrapolating from these network states to generic states is reasonable.

For the synaptic weight matrix $W^{(k)} \in \mathbb{R}^{o_k i_k}$ of the k^{th} layer, let us choose a transformation $\phi^k(W^{(k)}) = D^{(k)} W^{(k)} (B^{(k)})^T \in \mathbb{R}^{o'_k \times i'_k}$ (from the original matrix to a bigger matrix with dimensionality $o'_k \times i'_k$) that preserves task error. We will take

$$B^{(k)} = [\mathbb{I}_{i_k}, 0_{i'_k - i_k}]$$
$$D^{(k)} = [\mathbb{I}_{o_k}, 0_{o'_k - o_k}].$$

For the input layer, we demand that $i'_1 = i_1$ so that the number of inputs is preserved and $B^{(1)}$ is the identity matrix. For the output (d^{th}) layer, we will similarly demand that $o'_d = o_d$ so that the output dimensionality of the network is preserved. Intermediate layers in the transformed network have, by the above construction, greater (or equal) numbers of neurons as compared to the nominal network. Regardless of the nonlinearity σ (which may not satisfy $\sigma(0) = 0$, and indeed does not for sigmoidal nonlinearities as used in numerical examples of the paper), this transformation satisfies

$$F'[\phi(W)] = F[W].$$

Here we have essentially added synapses to the network with weight zero, in such a way that they contribute nothing to the input-output properties of the network. However as soon as their weights deviate from zero, this is no longer the case. So they do contribute to the gradient $\nabla F'[\phi(W)]$. If we represent the concatenated weight matrices of the network as a long vector $\mathbf{w}$, the described transformation can be effected by a function $\phi(\mathbf{w}) = A\mathbf{w} \in \mathbb{R}^{\tilde{N}}$, where $\tilde{N}$ is the number of weights in the expanded network, and where the matrix A satisfies $A^T A = \mathbb{I}_N$. Using the chain rule, we have, for any $\mathbf{w}^* \in \mathbb{R}^N$:

$$A^T \nabla F'[\phi(\mathbf{w}^*)] = \nabla F[\mathbf{w}^*] \tag{S.21a}$$
$$A^T \nabla^2 F'[\phi(\mathbf{w}^*)] A = \nabla^2 F[\mathbf{w}^*]. \tag{S.21b}$$

We will use equations Eq. (S.21) to somewhat constrain the learning rate. First note that

$$\|\nabla F'[\phi(\mathbf{w}^*)]\|_2 \geq \|\nabla F[\mathbf{w}^*]\|_2 \tag{S.22a}$$

through equation Eq. (S.21a) and semi-orthogonality of A. Heuristically, we can say that

$$\frac{\|\nabla F'[\phi(\mathbf{w}^*)]\|_2^2}{\|\nabla F[\mathbf{w}^*]\|_2^2} \approx \frac{\tilde{N}}{N}. \tag{S.22b}$$

This would be true if $\nabla F'[\phi(\mathbf{w}^*)]$ projected onto each of the eigenvectors of AA^T equally, given that AA^T has N eigenvalues of magnitude one, and $\tilde{N} - N$ of magnitude zero. In the same spirit, we take the heuristic

$$Tr(A^T \nabla^2 F'[\phi(\mathbf{w}^*)] A) \equiv Tr(\nabla^2 F'[\phi(\mathbf{w}^*)] AA^T) \approx \frac{\tilde{N}}{N} Tr(\nabla^2 F[\mathbf{w}^*]). \tag{S.22c}$$

The approximation similarly holds if the sum of the columns of $\nabla^2 F'[\phi(\mathbf{w}^*)]$ projects equally onto each of the eigenvectors of AA^T. Note that this approximation would not be reasonable in the previous linear example with a quadratic error function. In that case $\nabla^2 F'$ was constant over learning. Since A is also constant, if one encountered a situation where e.g. $\nabla^2 F'$ projected much more strongly onto eigenvectors of AA^T with nonzero eigenvalues, then that situation would hold over the entire course of learning, since it related two constant matrices. In the current, nonlinear example, by contrast, the error function is non-quadratic, and thus $\nabla^2 F'$ changes over time. It has columns that continually shift direction over the course of learning. On average, it is therefore reasonable to assume that their sum projects onto AA^T in the manner hypothesised.

Finally, expansion of equations Eq. (S.21) gives

$$\left\langle \nabla F[\mathbf{w}^*], \nabla^2 F[\mathbf{w}^*] \nabla F[\mathbf{w}^*] \right\rangle = \left\langle A^T \nabla F'[\phi(\mathbf{w}^*)], A^T \nabla^2 F'[\phi(\mathbf{w}^*)] AA^T \nabla F'[\phi(\mathbf{w}^*)] \right\rangle. \tag{S.22d}$$
$$\approx \left(\frac{N}{\tilde{N}}\right)^2 \left\langle \nabla F'[\phi(\mathbf{w}^*)], \nabla^2 F'[\phi(\mathbf{w}^*)] \nabla F'[\phi(\mathbf{w}^*)] \right\rangle, \tag{S.22e}$$

where we again assume uniformity of projection of $\nabla F'$ onto the eigenalues of AA^T to make the approximation. Normalising the LHS and RHS of the above equation by $\|\nabla F[\mathbf{w}^*]\|_2^2$ and $\|\nabla F'[\phi(\mathbf{w}^*)]\|_2^2$ respectively, equation Eq. (S.22b) implies

$$\left\langle \nabla \hat{F}[\mathbf{w}^*], \nabla^2 F[\mathbf{w}^*] \nabla \hat{F}[\mathbf{w}^*] \right\rangle \approx \left\langle \nabla \hat{F}'[\phi(\mathbf{w}^*)], \nabla^2 F'[\phi(\mathbf{w}^*)] \nabla \hat{F}'[\phi(\mathbf{w}^*)] \right\rangle. \tag{S.22f}$$

This, along with Eq. (S.22c), allows us to estimate how $\mathbf{G}^1_F$ grows with network size. We can then use equations Eq. (S.22) to estimate how local task difficulty varies with the number of synapses. Given N synapses in the original network, and $\tilde{N}$ in the expanded network, we get

$$\mathbb{E}\left[\mathbf{G}'_{F'}[\phi(\hat{\dot{\boldsymbol{\omega}}}_T)]\right] \|\phi(\dot{\boldsymbol{\omega}}_T)\|_2^2 \approx \sqrt{\frac{N}{\tilde{N}}} \gamma_1^2 \mathbf{G}^1_F[\mathbf{w}(0)] + \frac{\sqrt{\tilde{N}} Tr(\nabla^2 F[\mathbf{w}(0)])}{2\sqrt{N} \|\nabla F[\mathbf{w}(0)]\|_2} \left[\frac{\gamma_2^2}{\tilde{N}} + \frac{\gamma_3^2}{T}\right],$$

where $\mathbf{G'}_{F'}$ is the local task difficulty associated with the direction $\phi(\dot{\hat{\boldsymbol{\omega}}}_T)$ in the expanded network, given an initial synaptic configuration of $\phi[\mathbf{w}(0)]$. As we did in the linear case (see equation Eq. (S.20)), we can differentiate the expression for $\boldsymbol{\delta}$ in $\tilde{N}$, to find an N^* that optimises learning rate. As in equation Eq. (S.19), we take the approximation $C \approx \frac{1}{N^*}$ (justified subsequently). We get

$$N^* \approx \frac{T\gamma_2^2}{\gamma_3^2}\left[CN\frac{\gamma_1^2}{\gamma_2^2}+1\right], \tag{S.23a}$$

$$\approx \frac{T\gamma_2^2}{\gamma_3^2}\left[\frac{\gamma_1^2}{\gamma_2^2}\frac{N}{N^*}+1\right]. \tag{S.23b}$$

where N is the number of synapses in the original network, and C is as defined in equation Eq. (S.19). Note that N^* has a unique value: it can be solved as a quadratic with a single non-negative real root. Since $\boldsymbol{\delta}$ is monotonically increasing for $\tilde{N} \geq N^*$, and monotonically decreasing for $\tilde{N} \leq N^*$, we are assured that N^* is a global minimum of $\boldsymbol{\delta}$ over the domain $\tilde{N} \geq 0$.

As in equation Eq. (S.19), we took the approximation $C \approx \frac{1}{N^*}$. We therefore assume that $\nabla F'[\phi(\mathbf{w})]$ projects evenly, in expectation, onto the eigenvectors of $\nabla^2 F'[\phi(\mathbf{w})]$. In the linear case, this was reasonable as the Hessian was static. In this case, that is no longer true and a longer argument is necessary:

First let us formulate a notion of independence for matrices and vectors. Consider a generic matrix $M(t) \in \mathbb{R}^{N\times N}$ and a generic vector $r(t) \in \mathbb{R}^N$. We shall denote the i^{th} eigenvector/eigenvalue pair of $M(t)$ by $v_i(t)$ and $\lambda_i(t)$ respectively. We can write

$$M(t)r(t) = \sum_{i=1}^{N} c_i(t)\lambda_i(t)v_i(t),$$

and we say that $M(t)$ is independent of $r(t)$, over time, if

$$\mathbb{E}[c_i(t)\lambda_i(t)] = \mathbb{E}[c_i(t)]\mathbb{E}[\lambda_i(t)] \ \forall i,$$

where expectation is taken over time. For our heuristic $C = \frac{1}{N^*}$ to hold, we need $\nabla\hat{F}[\mathbf{w}(t)]$ to be independent of $\nabla^2 F[\mathbf{w}(t)]$ over time.

Now consider the nominal network. Equation Eq. (S.21) shows that the projection of $\nabla\hat{F}[\mathbf{w}(t)]$ onto the eigenvectors of $\nabla^2 F[\mathbf{w}(t)]$ is equal to the projection of $A^T\nabla F'[\phi(\mathbf{w}(t))]$ onto the latter. Therefore if A^T is independent (in the sense described above) of both $\nabla F'[\phi(\mathbf{w}(t))]$ and $\nabla^2 F[\mathbf{w}(t)]$, then we get independence of $\nabla\hat{F}[\mathbf{w}(t)]$ from $\nabla^2 F[\mathbf{w}(t)]$. Now A^T is a fixed matrix that is independent of the task, and a property of the network architecture only. So the independence assumption seems reasonable, thus justifying our approximation of C. Our heuristic for C leads to Eq. (S.23b), which can be expressed as a cubic equation in N^* with a single positive root, and thus solved. We have now considered a multilayer feedforward network with static nonlinearities at each neuron. We have constructively showed how any state of this layer can be embedded in a higher-dimensional layer in a manner that quantifiably lowers task difficulty.

One issue with the analysis is that we only consider the task difficulty of weight vectors in the image of ϕ (for the expanded network). This is a low-dimensional subset of the overall set of attainable weight vectors. For the analysis to be valid, we must assume that the expected local task difficulty is not much lower for weights in the image of ϕ, than for other weights with the same task error. If this were not true, then the local learning rate of the expanded network would jump whenever the weight vector of the network was in the image of ϕ. Given that ϕ is constructed with respect to the network architecture, but not the specific task being learnt, we find this unlikely in areas of weight space close to the image of the mapping $\phi(\mathcal{W})$ (where $\mathcal{W}$ denotes the space of small network weights). We could however imagine entirely 'new' areas of weight space emerging in a larger network, which exhibited network behaviours not realisable from the small network, and in which task difficulty was lower than comparable weights with similar values of $F[\mathbf{w}]$ in the image of ϕ. This would be a benefit of larger networks due to the more complex set of input-output behaviours that a larger network can exhibit, which we do not quantify here, and which is highly task-dependent. If significant, it would increase the true value of N^* relative to our estimate.

The mathematical analysis we conducted can be used as a template to conduct size-increasing transformations on more general neural network architectures. Suppose we have a generic neural network with synaptic weights $\mathbf{w}$. Suppose we add neurons to this generic neural network, and can fix the synaptic weights of the added neurons in such a way that the input-output properties of the network do not change. So the expanded network has weights $[\mathbf{w}, \tilde{\mathbf{w}}]$. Then the size-expanding neural network transformation can be expressed as a function ϕ, with $\phi(\mathbf{w}) = [\mathbf{w}, \tilde{\mathbf{w}}]$. Since the input-output properties of the network do not change, we get

$$F'[\phi(\mathbf{w})] = F[\mathbf{w}].$$

Since $\phi(\mathbf{w}) = [\mathbf{w}, \tilde{\mathbf{w}}]$, we get

$$\big(\nabla\phi(\mathbf{w})\big)^T\big(\nabla\phi(\mathbf{w})\big) = \mathbb{I}_N.$$

At any particular weight configuration $\mathbf{w}^*$, we can locally define $\nabla\phi(w) = A\mathbf{w} + B$, for some B. Note that the above equation implies semi-orthogonality of A, which allows us to repeat the original analysis of the current section.

Online learning and generalization error

Biologically plausible learners must learn features attributable to (near-) infinite input examples. However, they gain information on features from only single, or few, examples on each learning cycle. This introduces task-irrelevant plasticity, with magnitude dependent on how poorly the examples represent the 'average' example. In this section we show how such task-irrelevant plasticity fits into the γ_2 component (learning rule noise) of the plasticity decomposition described in the main paper. This allows us to use the results of the paper to make statements on the relationship between network size, and the generalizability of learning from small numbers of examples.

We consider a situation in which the learner can receive a (possibly infinite) set $\mathcal{U}$ of input vectors. There is some probability distribution $\mathbb{P}[u]$ specifying the probability density of drawing a particular input. The learner receives and processes an input u with input-dependent error $F[\mathbf{w}, u]$. We could think of a classification task as an example, where $F[\mathbf{w}, u]$ was the classification error associated with some input u to be classified.

In the above case, we are interested in the expected error of the learner on the next input drawn. This is often known as the generalization error. Mathematically, we have

$$F[\mathbf{w}] = \mathbb{E}_{\mathbb{P}}[F[\mathbf{w}, u]] := \int_U F[\mathbf{w}, u]\mathbb{P}[u]\ du. \tag{S.24}$$

The learner's objective is to minimise the generalization error. However, at a given time (e.g. $T = 0$), the learner receives a noisy approximation of $F[\mathbf{w}(0), u]$, rather than $F[\mathbf{w}(0)]$. How does this fit into our model of weight updates during learning, where we specify the weight update over the interval $[0, T]$ to be:

$$\dot{\boldsymbol{\omega}}_T = -\gamma_1 \nabla \hat{F}[\mathbf{w}(0)] + \gamma_2 \hat{\mathbf{n}}_2 + \gamma_3 \sqrt{\frac{N}{T}} \hat{\mathbf{n}}_3 ? \tag{S.25}$$

First let us write an equation analogous to the above, but using $F[\mathbf{w}, u]$ instead of $F[\mathbf{w}]$. We get

$$\dot{\boldsymbol{\omega}}_T = -\tilde{\gamma}_1 \nabla \hat{F}[\mathbf{w}(0), u] + \tilde{\gamma}_2 \tilde{\hat{\mathbf{n}}}_2 + \tilde{\gamma}_3 \sqrt{\frac{N}{T}} \hat{\mathbf{n}}_3. \tag{S.26}$$

Next, let us write $\nabla F[\mathbf{w}(0), u]$ in terms of $\nabla F[\mathbf{w}(0)]$. First note that by differentiating, we have

$$\mathbb{E}_{\mathbb{P}}[\nabla F[\mathbf{w}(0), u]] = \nabla F[\mathbf{w}(0)].$$

Therefore we can write

$$\nabla F[\mathbf{w}(0), u] = \nabla F[\mathbf{w}(0)] + \mathbf{r}, \tag{S.27}$$

where

$$\mathbb{E}[\mathbf{r}] = 0 \tag{S.28a}$$

$$\|\mathbf{r}\|_2^2 = \|\nabla F[\mathbf{w}(0), u] - \nabla F[\mathbf{w}(0)]\|_2^2 \tag{S.28b}$$

$$\mathbb{E}[\|\mathbf{r}\|_2^2] = Tr\Big[Cov\left(\nabla F[\mathbf{w}(0), u] - \nabla F[\mathbf{w}(0)]\right)\Big] \tag{S.28c}$$

This allows us to write equation Eq. (S.26) in terms of Eq. (S.25). We get

$$\dot{\boldsymbol{\omega}}_T = -\tilde{\gamma}_1 \frac{\|\nabla F[\mathbf{w}(0)]\|_2}{\|\nabla F[\mathbf{w}(0), u]\|_2} \nabla \hat{F}[\mathbf{w}(0)] - \frac{\tilde{\gamma}_1}{\|\nabla F[\mathbf{w}(0), u]\|_2} \mathbf{r} + \tilde{\gamma}_2 \tilde{\hat{\mathbf{n}}}_2 + \tilde{\gamma}_3 \sqrt{\frac{N}{T}} \hat{\mathbf{n}}_3.$$

The above equation can be written in the form of equation Eq. (S.25). We only need to make the following substitutions:

$$\mathbf{n}_2 = \tilde{\mathbf{n}}_2 + \mathbf{r} \tag{S.29a}$$

$$\gamma_1 = \tilde{\gamma}_1 \frac{\|\nabla F[\mathbf{w}(0)]\|_2}{\|\nabla F[\mathbf{w}(0), u]\|_2} \tag{S.29b}$$

$$\gamma_2 = \sqrt{\tilde{\gamma}_2^2 + \frac{\tilde{\gamma}_1^2 \|\mathbf{r}\|_2^2}{\|\nabla F[\mathbf{w}(0), u]\|_2^2}}. \tag{S.29c}$$

Critically, we have that if $\tilde{\hat{\mathbf{n}}}_2$ obeys uncorrelatedness (in expectation) with the gradient $\nabla[F[\mathbf{w}(0)]]$, so too does $\hat{\mathbf{n}}_2$. So the assumptions previously placed on the $\hat{\mathbf{n}}_2$ and $\hat{\mathbf{n}}_3$ terms (i.e. equations Eq. (S.6b) and Eq. (S.10)) continue to hold. Note that

$$\mathbb{E}[\gamma_2] = \sqrt{\tilde{\gamma}_2^2 + \frac{\tilde{\gamma}_1^2 \|\mathbf{r}\|_2^2}{\|\mathbf{r}\|_2^2 + \|\nabla F[\mathbf{w}(0)]\|_2^2}}.$$

This makes clear that $\mathbb{E}[\gamma_2]$ will not change under a network size expansion of the form described in SI section *Learning in a nonlinear, feedforward network*, as long as $\frac{\|\mathbf{r}\|_2^2}{\|\nabla F[\mathbf{w}(0)]\|_2^2}$ remains invariant. Thus task-irrelevant plasticity depends on the relative error of $\nabla F[\mathbf{w}(0), u]$ in approximating $\nabla F[\mathbf{w}(0)]$. We know, from the analysis of equations Eq. (S.22), that $\|\nabla F[\mathbf{w}(0)]\|_2$ grows linearly with network size. We can apply the same analysis to $\|\mathbf{r}\|_2$ (instead of $\|F[\mathbf{w}(0)]\|_2$), and conclude the same linear growth with network size. So their ratio is preserved, and $\mathbb{E}[\gamma_2]$ does not change with network size.

In summary, we can transfer the qualitative results of the paper to the above online learning scenario, where minimization of generalization error is the objective. Gradient noise due to approximating $F[\mathbf{w}(0)]$ with its stochastic analogue $F[\mathbf{w}(0), u]$ has the effect of increasing task-irrelevant plasticity (i.e. γ_2) by an amount dependent on the trace of the covariance matrix of $\nabla[F[\mathbf{w}(0), u]] - \nabla F[\mathbf{w}(0)]$ (as we see from equations Eq. (S.28c) and Eq. (S.29c)). We showed that the magnitude of increase in γ_2 does not depend on network size. Therefore bigger networks will learn better in its presence, since the main paper shows that size expansions benefit learning rate and steady state error, for fixed γ_1 and γ_2 (with $\gamma_3 = 0$). This is demonstrated in Figure S1, where $\mathbb{P}[u]$ is a Gaussian distribution, two sizes of network are trained sequentially on inputs $u \in \mathcal{U}$, and their generalisation errors are compared. It is also demonstrated in Figure 2C of the main paper, where students of many different sizes learn in the same online setting. Our results predict that excessively large networks in this setting are penalised when $\gamma_3 > 0$, and this is also demonstrated in Figure 2D. We see that the qualitative, non-monotonic relationship between network size and generalization error holds. However, we cannot calculate γ_2 in this online setting due to the unquantifiable r term in Eq. (S.29c). Therefore we can qualitatively, but not quantitatively validate results (i.e. we cannot find an optimal network size).

There may be additional advantages to the generalization error attainable by very large networks within the biologically plausible context of online learning not restricted to a training set. These relate to favourable properties of the statistical distribution of synaptic weights achieved by large networks. Promising initial work (e.g. (2)) has been made in this direction. Note however, that the conclusions of the latter paper are not directly applicable in our context: they deal with the case that $\gamma_2 = \gamma_3 = 0$, i.e. the effect of network size on learning performance when the network learns by batch gradient descent. In contrast, we deal with how network size affects the impact of nonzero γ_2 and γ_3 on learning.

Regularization and generalization error

In the previous section, as in the main paper, we dealt with an online learner receiving inputs sequentially, from some underlying probability distribution. In machine learning, neural networks often learn in a different scenario, where learning is conducted with respect to a finite training set of data. Nevertheless, generalization error on unseen inputs, rather than training error on the training set, is the desired cost function to be minimized. This machine learning scenario might be relevant in certain biological contexts. One could imagine memorized inputs (an effective training set) being continually replayed (without degradation), and learning occuring with respect to these. In this section we consider the predictions that our paper makes in the event that learning is conducted on a finite, restricted training set, but performance is evaluated on inputs not belonging to this training set.

We first show that the bias induced by learning from a finite training set cannot be considered as purely task-irrelevant plasticity (as defined in this paper) relative to the generalization error. This shows why various forms of regularization can improve generalization error when learning from a training set. This stands in contrast to the intrinsic noise we defined in this paper, which degrades steady state error.

We next show that the results of our paper corroborate and give analytic insight into the well-known heuristic that larger networks benefit from regularization/weight noise, whereas smaller networks are hindered by it.

Finally we consider a hypothetical plasticity source that stochastically implements $\mathcal{L}_2$ regularization (i.e. weight decay) during learning. We investigate how task-irrelevant plasticity induced by such a term changes with network size. We see that it does not grow with network size, and as such can be considered as a γ_2 component of synaptic plasticity, but not a γ_3 component. The analysis used here can be modified to investigate the relationship between network size and the effect of stochastic plasticity terms implementing different types of regularization.

As in the previous section, we consider a learner that receives input vectors u from some set $\mathcal{U}$, according to a probability density $\mathbb{P}[u]$. The task is to find weights that minimise the generalization error, (see equation Eq. (S.24)). In artificial settings, drawing u from the entire set $\mathcal{U}$ is often not practical. One common reason is that inputs often have to be preprocessed (e.g. labelled with optimal outputs for supervised learning) in order for a gradient $\nabla F[\mathbf{w}, u]$ to be calculable.

To deal with these practical issues, a subset $\mathcal{U}_{\text{train}}$ of inputs known as the *training set* is drawn according to $\mathbb{P}[u]$. Every input $u \in \mathcal{U}_{\text{train}}$ is preprocessed (e.g. labelled) appropriately, and training is only conducted with respect to inputs in $\mathcal{U}_{\text{train}}$. In order to test generalization error, a further test set $\mathcal{U}_{\text{test}}$ may be additionally drawn, but not used for training.

In this setting, we can define the training error as

$$F_{\text{train}}[\mathbf{w}] = \int_{\mathcal{U}_{\text{train}}} F[\mathbf{w}, u]\mathbb{P}[u|u \in \mathcal{U}_{\text{train}}]\ du.$$

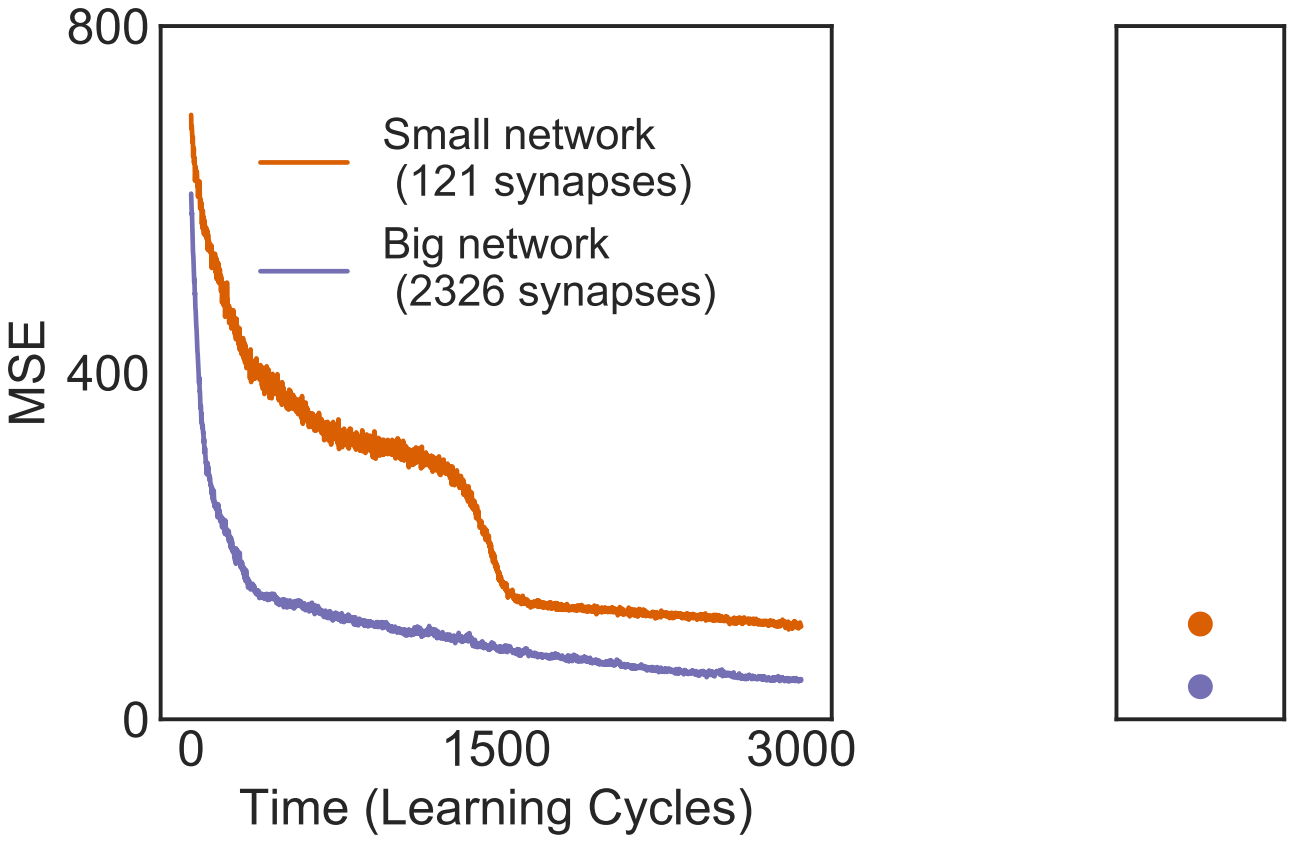

Fig. S1. Dynamics of learning in a situation where the true task-error gradient cannot be computed, thus adding noise to the learning rule. Comparison of big and small networks. Both networks have the same number of inputs (10) and outputs (10), and learn the same fixed mapping from a teacher network. Over each learning cycle, a single input is drawn from a unit Gaussian distribution. The gradient of task error is calculated only with respect to this input (i.e. stochastic gradient descent). No extra task irrelevant plasticity is added. The task error is estimated by evaluating task error over 1000 Gaussian inputs. **Left**: Estimated task error is plotted over time for the big and small networks. **Right**: Mean steady state error after 3000 learning cycles, over 12 iterations. Error bars are invisible due to their small size: A single standard deviation from the mean is $[1.67858973, 1.0943906]$ for the small and big networks respectively.

Now at each learning cycle, the learner receives an input $u \in \mathcal{U}_{\text{train}}$, drawn from the probability density $\mathbb{P}[u|u \in \mathcal{U}_{\text{train}}]$. So we have

$$\mathbb{E}[\nabla F[\mathbf{w}, u]] = \nabla F_{\text{train}}[\mathbf{w}].$$

This means we can follow the mathematics of the previous section *Online learning and generalization error*, but replacing F with F_{train}, and arrive at an update rule

$$\dot{\boldsymbol{\omega}}_T = -\gamma_1 \nabla \hat{F}_{\text{train}}[\mathbf{w}(0)] + \gamma_2 \hat{\mathbf{n}}_2 + \gamma_3 \sqrt{\frac{N}{T}} \hat{\mathbf{n}}_3, \tag{S.30}$$

where

$$\mathbb{E}[\langle \nabla F_{\text{train}}[\mathbf{w}(0)], \hat{\mathbf{n}}_2 \rangle] = 0.$$

The results of the paper can now be applied directly to the training error F_{train} (as opposed to the generalization error F). Let us set $\gamma_3 = 0$ to reflect the fact that intrinsic synaptic noise is avoidable in artificial neural networks. In this case, the results of the paper imply that steady state training error should decrease with network size. This is well known. Our results show that it is not necessarily due to any fundamental incapability of a small network to perfectly learn the task, but rather due to bigger networks dealing better with task-irrelevant plasticity (as comes from e.g. using a stochastic gradient).

What about generalization error? If we reformulate equation Eq. (S.30) so that the γ_1 coefficient is the generalization error F, we arrive at

$$\dot{\boldsymbol{\omega}}_T = -\gamma_1 \nabla \hat{F}[\mathbf{w}(0)] + \gamma_2 \left[\frac{-\gamma_1}{\gamma_2} (\nabla \hat{F}_{\text{train}}[\mathbf{w}(0)] - \nabla \hat{F}[\mathbf{w}(0)]) + \hat{\mathbf{n}}_2 \right] + \gamma_3 \sqrt{\frac{N}{T}} \hat{\mathbf{n}}_3.$$

The portion of the above equation that is square bracketed now represents the task-irrelevant plasticity term. But it is not uncorrelated, in expectation, with $\nabla F[\mathbf{w}(0)]$, and therefore the results of the paper cannot be applied. Indeed it is positively correlated with $\nabla F[\mathbf{w}(0)]$ unless $\nabla F_{\text{train}}[\mathbf{w}] = \nabla F[\mathbf{w}]$ is satisfied. This positive correlation becomes large as we approach a local minimum of $F_{\text{train}}[\mathbf{w}]$ (i.e. $\|\nabla F_{\text{train}}[\mathbf{w}]\|_2$ approaches zero). This reflects the fact that task-relevant plasticity for F_{train} can increase $F[\mathbf{w}]$, when the former is close to a stationary point, a phenomenon known as overtraining.

Let us further consider generalization error in the artificial setting just described, and in particular overtraining. Training over time often exhibits the qualitative relationship depicted in Figure S2. While initial improvements in training error improve generalization error, at some critical point further improvements in training error degrade generalization error. This is the point at which overtraining starts. If we take for granted the qualitative relationship shown in Figure S2, then we need a mechanism to prevent overtraining in large networks (where our results indicate that steady-state training error will be low). One way is to terminate learning once the training error reaches a cutoff. Indeed, a body of research (see e.g. (2, 3)) focuses on the optimal stopping time for gradient descent learning, and its relationship to network size.

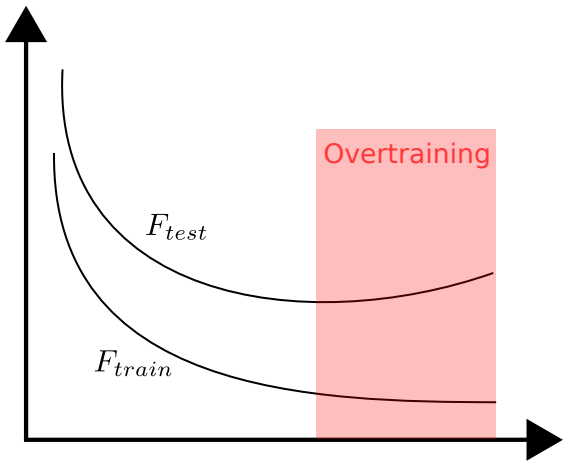

Fig. S2. Learning from a training set. As training error decreases so to does generalization error. Past a point, however, further decreases in training error actually increase generalization error. This is the phenomenon of overfitting.

Another way is to add some form of intrinsic noise to the weights of the network during training, corresponding to increasing the γ_2 and/or γ_3 terms in the language of this paper. Our paper predicts that such noise will increase training error. Meanwhile Figure S2 predicts that increased training error will be beneficial for generalization error. Indeed, this is known to be an effective method of improving generalization error. Techniques such as dropout (4), and the stochastic delta rule (5) all noisily perturb weights of the network. Results from (4) suggest that this weight perturbation increases training error, but decreases generalization error, as we expect.

We have provided qualitative insight into why adding weight noise can help generalization error at the cost of increasing training error. A comparison of the extent to which different types of weight noise aid generalization in machine learning is outside the scope of this paper. However, a relevant question is whether such terms contribute intrinsic synaptic noise to the generalization error.

An answer to this question depends upon the specific type of weight noise applied. Nevertheless, we will provide a template for such an analysis, using stochastic $\mathcal{L}_2$ regularization as a specific example. We find that a stochastic $\mathcal{L}_2$ regularization term can only contribute γ_2 plasticity, rather than γ_3 plasticity (i.e. intrinsic synaptic noise). So one can apply it to an arbitrarily large network, without expecting an eventual increase in generalization error.

Let us first define the $\mathcal{L}_2$ regularizing function:

$$R[\mathbf{w}] = c\|\mathbf{w}\|_2^2,$$

for some arbitrary, fixed, choice of hyperparameter $c > 0$. Let us consider a neural network adapting through a combination of gradient descent, and a stochastic regularization term $\epsilon(\mathbf{w})$. Specifically,

$$F[\mathbf{w}] = -\gamma_1 \nabla F[\mathbf{w}] + \epsilon(\mathbf{w}),$$
$$\mathbb{E}[\epsilon(\mathbf{w})] := D(\mathbf{w}).$$

Now $D(\mathbf{w})$ must descend the gradient of $R[\mathbf{w}]$. Therefore we have

$$D(\mathbf{w}) = -\nabla R[\mathbf{w}] = -2c\mathbf{w}.$$

Now let us consider a worst-case scenario in which all plasticity due to $D(\mathbf{w})$ is task-irrelevant. Mathematically,

$$\langle D(\mathbf{w}), \nabla F[\mathbf{w}] \rangle = 0.$$

How does the task-irrelevant plasticity, which is now equal in magnitude to $\|D(\mathbf{w})\|_2$, grow with network size?

Recall that the network expansion increasing network size (nonlinear case) can be represented by a function $\phi : \mathbb{R}^N \to \mathbb{R}^{\tilde{N}}$, transporting the weights of the smaller network to the larger network (see SI section: *Learning in a nonlinear, feedforward network*). If we represent the error function of the larger network by F', we have that

$$F'[\phi(\mathbf{w})] = F[\mathbf{w}],$$

as the network expansion preserves input-output behaviour. Locally around $\mathbf{w}$ we also have

$$\phi(\mathbf{w}) = A\mathbf{w},$$

where A is a semi-orthogonal matrix (i.e. $A^T A = \mathbb{I}$). Critically, note that the network expansion **preserves the $\mathcal{L}_2$ norm of the weights**, as

$$\|\mathbf{w}\|_2^2 = \|\mathbf{w}^T A^T A\mathbf{w}\|_2^2 = \|\phi(\mathbf{w})\|_2^2.$$

Therefore, we have that $R[\mathbf{w}] = R'[\phi(\mathbf{w})]$, where R' represents the image of R under the expansion ϕ. Next, how does $\nabla R[\mathbf{w}]$ change under this expansion? Explicit differentiation gives

$$\begin{aligned}
&\nabla R'[\phi(\mathbf{w})] = -2c\phi(\mathbf{w}) = -2cA\mathbf{w}, \\
&\Rightarrow \|\nabla R'[\phi(\mathbf{w})]\|_2 = \|\nabla R[\mathbf{w}]\|_2, \\
&\Rightarrow \|D'[\phi(\mathbf{w})]\|_2 = \|D[\mathbf{w}]\|_2.
\end{aligned}$$

So we see that the magnitude of D remains unchanged under the transformation ϕ. Therefore it is independent of the network size, and constitutes a source of γ_2 plasticity.

As an aside, we previously showed (see SI section *Learning in a nonlinear,feedforward network*) that the network expansion ϕ will on average increase $\|\nabla F'[\phi(\mathbf{w})]\|_2$ at a rate proportional to $\sqrt{\frac{\tilde{N}}{N}}$ over a weight trajectory during learning. However, $\|\nabla R[\mathbf{w}]\|_2$ does not grow with $\sqrt{\tilde{N}}$ for the special case of $R[\mathbf{w}] = \|\mathbf{w}\|_2^2$. This is consistent with the result in the paper for the following reason:

In the main paper we used the following chain rule identity to see how $\|\nabla F'[\phi(\mathbf{w})]\|_2$ depended on the size of the expanded network:

$$A^T \nabla F'[\phi(\mathbf{w})] = \nabla F[\mathbf{w}].$$

(see SI section: *Learning in a nonlinear, feedforward, neural network*). If $\nabla F'[\phi(\mathbf{w})]$ projected approximately equally onto the different eigenvectors of A, then the following would hold:

$$\frac{\|\nabla F'[\phi(\mathbf{w})]\|_2}{\|\nabla F[\mathbf{w}]\|_2} \approx \sqrt{\frac{\tilde{N}}{N}}.$$

We claimed that this equality of projection was a reasonable assumption in the general case, since the direction $\nabla F[\mathbf{w}]$ would constantly be changing, and the matrix A was constructed independently of the function $F[\mathbf{w}]$. This assumption is **invalid** when considering $R[\mathbf{w}]$ instead of $F[\mathbf{w}]$. Consequently, we cannot use a similar line of argument to determine how $\|\nabla R'[\phi(\mathbf{w})]\|_2$ grows with $\tilde{N}$. Why? Explicit differentiation gives

$$\nabla R'[\phi(\mathbf{w})] = 2A\mathbf{w}.$$

This is contained within the image of A, and so is orthogonal to any of the eigenvectors of A with zero eigenvalue. It only projects onto eigenvectors of A with eigenvalue 1.

Dropout is a common form of stochastic regularization in artificial neural networks (4). It is hard to determine the expected effect of dropout on synaptic plasticity, however. This quantity corresponds to $D(\mathbf{w})$ as just defined in the context of stochastic $\mathcal{L}_2$ regularization. In special cases dropout is known to be equal to or approximately equal (to first-order) to stochastic weighted $\mathcal{L}_2$ regularization (see e.g. (6, 7)). In such cases, the analysis above shows that dropout does not correspond to intrinsic synaptic noise.

More generally, we numerically estimated how task-irrelevant plasticity due to dropout depends on network size in nonlinear feedforward networks using a student-teacher task. Figure S3 shows the magnitude of task-irrelevant plasticity due to dropout as a function of network size. For comparison, we plot the total contribution of intrinsic synaptic noise, $A_0\sqrt{N}$, where A_0 is chosen so that the contributions are equal for the smallest network simulated. The simulations indicate that total contribution due to dropout grows at a much lower rate than intrinsic noise. This implies that the per-synapse magnitude of task-irrelevant plasticity due to dropout decreases with network size, which distinguishes it from intrinsic synaptic noise. Moreover, for larger network sizes the magnitude of dropout-induced plasticity appears to saturate. If it does saturate, then dropout would constitute, at worst, a source of γ_2 plasticity. In this case dropout would not impose an optimal network size in the way that intrinsic synaptic noise does.

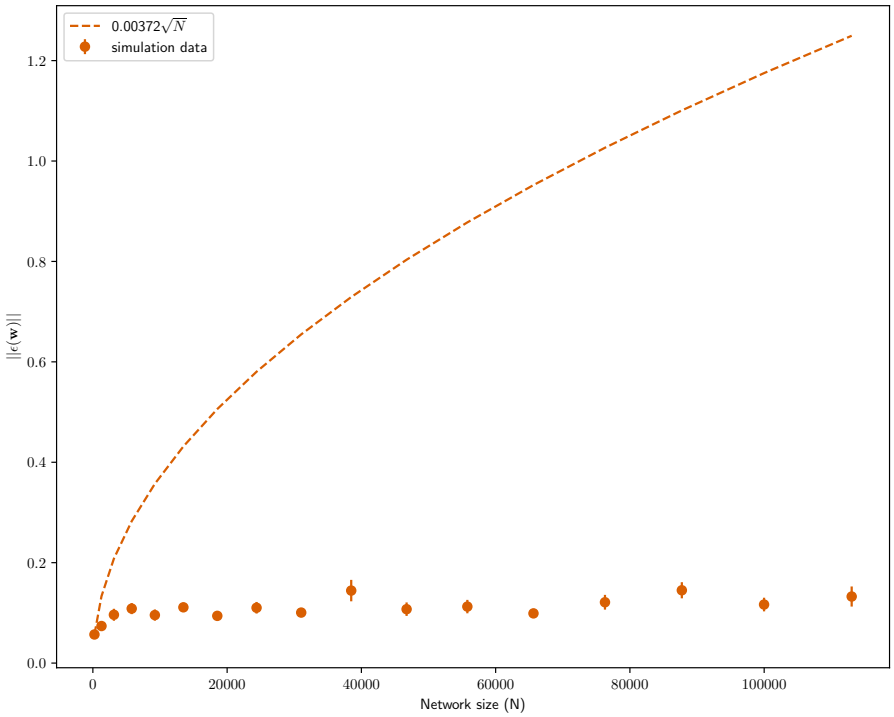

Fig. S3. Dropout in an nonlinear feedforward network is distinct from intrinsic noise. Plots show the dependence of steady-state task-irrelevant plasticity due to dropout (points) compared with intrinsic synaptic noise (dashed line) as a function of network size (N, number of synapses) in nonlinear feedforward neural networks. The curve corresponding to intrinsic noise is scaled to coincide with the first simulation datapoint for dropout noise (giving the coefficient 0.00372).

Simulation details: nonlinear, feedforward neural networks with 5 hidden layers were trained for 1000 training cycles on the same task, generated by a teacher network. The teacher network was fully connected with the following numbers of neurons in each layer $[10, 5, 5, 5, 5, 10]$. All student networks had ten inputs and ten outputs. Size increases were generated by increasing the number of hidden layer neurons from 5 to 165 neurons in each hidden layer. Weight initialisations were as described in the *Methods* section. Training implemented standard stochastic gradient descent with dropout, using a fixed learning rate of 0.02. Dropout probability was 20% at the input layer, and 50% at hidden layers. After training, we calculated weight updates in the presence and absence of dropout; $\epsilon(\mathbf{w})$ was taken as the difference between these two updates. $\|\epsilon(\mathbf{w})\|_2$ is therefore an upper bound on the magnitude of task-irrelevant plasticity induced by dropout. We plot $\|\epsilon(\mathbf{w})\|_2$ for different network sizes, averaged over 10 repeats. Error bars depict $\pm$ one standard error.

References

1. Bengio Y (2012) Practical recommendations for gradient-based training of deep architectures in *Neural networks: Tricks of the trade*. (Springer), pp. 437–478.
2. Advani MS, Saxe AM (2017) High-dimensional dynamics of generalization error in neural networks. *arXiv:1710.03667 [physics, q-bio, stat]*.
3. Changfeng Wang, Venkatesh S, Judd J (1995) Optimal stopping and effective machine complexity in learning in *Proceedings of 1995 IEEE International Symposium on Information Theory*. (IEEE, Whistler, BC, Canada), p. 169.
4. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

5. Hanson SJ (1990) A stochastic version of the delta rule. *Physica D: Nonlinear Phenomena* 42(1-3):265–272.
6. Wager S, Wang S, Liang PS (2013) Dropout training as adaptive regularization in *Advances in neural information processing systems*. pp. 351–359.
7. Baldi P, Sadowski PJ (2013) Understanding dropout in *Advances in neural information processing systems*. pp. 2814–2822.