Weight perturbation learning outperforms node perturbation on broad classes of temporally extended tasks

Paul Züge[a], Christian Klos[a], and Raoul-Martin Memmesheimer[a,✉]

[a]Neural Network Dynamics and Computation, Institute of Genetics, University of Bonn

Correspondence: ***rm.memmesheimer@uni-bonn.de***

Biological constraints often impose restrictions for plausible plasticity rules such as locality and reward-based rather than supervised learning. Two learning rules that comply with these restrictions are weight (WP) and node (NP) perturbation. NP is often used in learning studies, in particular as a benchmark; it is considered to be superior to WP and more likely neurobiologically realized, as the number of weights and therefore their perturbation dimension typically massively exceed the number of nodes. Here we show that this conclusion no longer holds when we take two biologically relevant properties into account: First, tasks extend in time. This increases the perturbation dimension of NP but not WP. Second, tasks are low dimensional, with many weight configurations providing solutions. We analytically delineate regimes where these properties let WP perform as well as or better than NP. Further we find qualitative features of the weight and error dynamics that allow to distinguish which of the rules underlie a learning process: in WP, but not NP, weights mediating zero input diffuse and gathering batches of subtasks in a trial decreases the number of trials required. The insights suggest new learning rules, which combine for specific task types the advantages of WP and NP. Using numerical simulations, we generalize the results to networks with various architectures solving biologically relevant and standard network learning tasks. Our findings suggest WP and NP as similarly plausible candidates for learning in the brain and as similarly important benchmarks.

neural networks | reinforcement learning | model-free | perturbation-based learning | weight perturbations

Statement of significance

Neural networks can learn by first perturbing the network weights or the activity of neurons and thereafter consolidating perturbations that improve the network performance. Weight perturbation learning is considered less efficient, useful and biologically plausible, because there are many more connection weights than neurons, such that generating beneficial perturbations seems less likely. We show that the argument does no longer hold when accounting for two features common in biology: tasks extend in time and the neuronal dynamics are low dimensional. In particular, we find that perturbing the weights is comparably good or better in various biologically relevant and standard network learning applications. This indicates that weight perturbation learning is similarly useful and a plausible candidate for learning in the brain.

Introduction

DIFFERENT, usually combined strategies underlie the learning of tasks in humans and other animals (1, 2). Supervised learning allows large, rapid improvements. It is based on observing in which way an action was erroneous and on the ability of the nervous system to use this information for the improvement of neuronal dynamics in a directed manner. This may be implemented by translating an error vector into a vector of suitable synaptic weight updates (3). Fast learning could be achieved by directly adapting the dynamics (4). Reward-based learning (reinforcement learning), in contrast, uses only a scalar feedback signal. It is thus also applicable if errors are known with little specificity, for example because there is only sparse, delayed feedback about the cumulative effect of actions, which might only tell whether an action was erroneous but not how the generating neural activity can be improved. Since only unspecific error information is used, finding the right direction of weight changes in the nervous system may require exhaustive exploration (2, 5). This complicates learning and slows it down. Still, reward-based learning allows animals to solve tasks as diverse as classical conditioning, birdsong learning, navigating through a maze, adaptation of movements towards a goal or gradual effort reduction in movements that already reach their goal (1, 2, 5–8).

A variety of models for reward-based learning have been developed in the context of theoretical neuroscience and machine learning (7, 8). Two conceptually straightforward implementations of such learning in neural networks are weight perturbation (WP) (9, 10) and node perturbation (NP) (11, 12). Their underlying idea is to add perturbations to the weights or to the summed weighted inputs and to correlate them to the change of task performance. If the reward increases due to an attempted perturbation, the weights or the node dynamics are changed in its direction. If the reward decreases, the changes are chosen oppositely. WP and NP are widely used models for reward-based learning in biological neural networks, due to four properties (4, 9, 11–16): (i) They are (with minor modifications) biologically plausible. (ii) They are applicable to a broad variety of networks and tasks. (iii) They are accessible to analytical exploration. (iv) They are optimal in the sense that the average of the generated weight change taken over all noise realizations

P.Z. performed the analytical research; C.K. and P.Z. performed the numerical research; P.Z., C.K. and R.-M.M. conceptualized the analytical and numerical research; R.-M.M., P.Z. and C.K. conceived the study; P.Z., R.-M.M. and C.K. wrote the manuscript. The authors declare no competing interests.

is along the reward gradient. The schemes' names were originally coined for approaches that directly estimate the individual components of the gradient using single perturbations to each weight or node (17).

WP explores with its random perturbations a space with dimensionality equal to the number of weights. For trials without temporal extent, NP only needs to explore a space with dimensionality equal to the number of nodes. The chain rule, amounting to a simple multiplication with the unweighted input strength, then allows to translate a desirable change in the summed weighted inputs into a change in a particular weight strength. NP thus uses additional information on the structure of the network (namely the linearity of input summation) to reduce the required exploration.

In linear approximation the optimal direction of weight changes aligns with the direction of the gradient of the reward. WP and NP seemingly attempt to find this direction by trying out random perturbations. Since the dimension of the space of possible perturbation directions is large, the probability of finding the gradient is small and a lot of exploration is necessary. This impedes WP and NP. The number of weights and thus the dimensionality of the perturbation space searched by WP is much larger than the number of nodes. NP is thus considered more suitable for reward-based neural network learning (3, 11, 12, 14, 18, 19) and its implementation in biological neural networks (15, 16). This is supported by quantitative analysis: (11) considered M linear perceptrons with N random inputs, using a student-teacher task. They found that for WP the optimal convergence rate of the student to the teacher weight matrix is by a factor NM worse than for exact gradient descent (GD). This is consistent with the argument that WP needs to search the NM-dimensional weight space to find the gradient, which is directly computed by GD. Accordingly, NP is worse than gradient descent by the dimensionality M of the node perturbation space.

The prerequisites of the arguments sketched above, however, do not hold in many biological situations. First, tasks in biology often extend in time and have a reward feedback that is temporally distant from the action (1, 6, 7, 12, 16). Second, the effective dimension of neural trajectories and of learning tasks is often comparably low (20–22). Our article analytically and numerically explores the perturbation-based learning of tasks with these features.

The article is structured as follows. First, we introduce the employed WP and NP learning models. We then derive analytic expressions for the evolution of expected error (negative reward) in linear networks solving temporally extended, low-dimensional reward-based learning tasks. This allows to identify conditions for which WP outperforms NP as well as the underlying reasons. Furthermore we delineate distinguishing qualitative characteristics of the weight and error dynamics. Finally we numerically show that WP is comparably good or outperforms NP in different biologically relevant and standard network learning applications.

Results

Learning models and task principles. Our study models the learning of tasks that are temporally extended. Time is split into discrete steps, indexed by $t = 1, \ldots, T$, where T is the duration of a trial. During this period, a neural network receives external input and generates output. At the end of a trial, it receives a scalar error feedback E about its performance (10, 12, 16, 18, 23). To quantitatively introduce the learning rules, we consider a neuron i, which may be part of a larger network. It generates in the t-th time bin an output firing rate z_{it}, in response to the firing rates r_{jt} of its N presynaptic neurons,

$$z_{it} = g\left(\sum_{j=1}^{N} w_{ij} r_{jt}\right). \tag{1}$$

Here w_{ij} is the weight of the synapse from neuron j to neuron i. The generally nonlinear activation function g implements the relation between the total input current and the output firing rate of the neuron (7, 24). We note that the individual synaptic input currents $w_{ij} r_{jt}$ in the model sum up linearly. This is a standard assumption, and it is a requirement for the NP scheme (11, 12, 17, 25). In presence of nonlinear dendritic compartments (26, 27), each of these could be an independently perturbed node.

We model WP learning by adding in the beginning of a trial a temporally static weight change ξ_{ij}^{WP} to each of the weights w_{ij} (10, 23). The output of the neuron then reads

$$z_{it}^{\mathrm{pert,WP}} = g\left(\sum_{j=1}^{N} (w_{ij} + \xi_{ij}^{\mathrm{WP}}) r_{jt}^{\mathrm{pert,WP}}\right), \tag{2}$$

where $r_{jt}^{\mathrm{pert,WP}}$ are the input rates, which may have a perturbation due to upstream perturbed weights. ξ_{ij}^{WP} are independent and identically distributed (iid) Gaussian white noise perturbations with standard deviation σ_{WP}, $\langle \xi_{ij}^{\mathrm{WP}} \xi_{kl}^{\mathrm{WP}} \rangle = \delta_{ik}\delta_{jl}\sigma_{\mathrm{WP}}^2$, where the angular brackets denote the average over perturbation realizations and δ is the Kronecker delta. The perturbations ξ_{ij}^{WP} change the output, which in turn influences the reward received at the end of the trial, Fig. 1a) left hand side. We usually assume that the difference in reward between the perturbed and an unperturbed trial with $\xi_{ij}^{\mathrm{WP}} = 0$ for all i, j is used to estimate the gradient: When the reward increases, for small perturbations this means that the tried perturbation has a positive component along the reward gradient. Consequently the update is chosen in the direction of that perturbation. When the reward decreases, the update is chosen in the opposite direction. We use the update rule

$$\Delta w_{ij}^{\mathrm{WP}} = -\frac{\eta}{\sigma_{\mathrm{WP}}^2}(E^{\mathrm{pert}} - E)\xi_{ij}^{\mathrm{WP}}, \tag{3}$$

where η is the learning rate, E^{pert} is the error of the perturbed trial and E is the error of the unperturbed trial. For the delayed non-match-to-sample (DNMS) task, the error of the unperturbed trial is replaced by an average over the previous errors for biological plausibility. The proportionality of update

size and obtained reward implies that when averaging over the distribution of the perturbations, the weight change

$$\langle \Delta w_{ij}^{\mathrm{WP}} \rangle \approx -\eta \frac{\partial E}{\partial w_{ij}} \tag{4}$$

is parallel to the reward gradient (Supplementary Note (SN) 1, Eq. (S2)). Since $\langle \xi_{ij}^{\mathrm{WP}} \rangle = 0$, this holds for any baseline in Eq. (3). The employed choice of baseline E guarantees that for small perturbations the weight change has a positive component in the direction of the reward gradient and thus always reduces the error for sufficiently small learning rate η (10). In fact it minimizes the update noise, i.e. the fluctuations of updates around the gradient (SN 1, Eq. (S5)).

WP treats the system as a black box, mapping parameters w onto a scalar error function E. In other words, it uses the information that the weights are fixed parameters during a trial, but does not take advantage of specifics of the network structure. This is in contrast to NP, which takes into account some minimal structural knowledge, namely the linear summation of input currents, but not the constancy of the weights.

Instead of perturbing the weights directly, NP adds noise to the sum of the inputs,

$$z_{it}^{\mathrm{pert,NP}} = g\left(\sum_{j=1}^{N} w_{ij} r_{jt}^{\mathrm{pert,NP}} + \xi_{it}^{\mathrm{NP}} \right) \tag{5}$$

(11, 12), see Figure 1a) right hand side. ξ_{it}^{NP} are iid Gaussian white noise perturbations with standard deviation σ_{NP}, $\langle \xi_{it}^{\mathrm{NP}} \xi_{ms}^{\mathrm{NP}} \rangle = \delta_{im}\delta_{ts}\sigma_{\mathrm{NP}}^2$. We note that for temporally extended tasks, in contrast to WP the noise must be time dependent to explore the space of time-dependent sums of inputs (12). For temporally constant noise, only the temporal mean of the total input would be varied and thus improved.

The NP update rule can be defined as

$$\Delta w_{ij}^{\mathrm{NP}} = -\frac{\eta}{\sigma_{\mathrm{NP}}^2} (E^{\mathrm{pert}} - E) \sum_{t=1}^{T} \xi_{it}^{\mathrm{NP}} r_{jt} \tag{6}$$

(12), with the eligibility trace $\sum_{t=1}^{T} \xi_{it}^{\mathrm{NP}} r_{jt}$. As for WP, this yields an average weight update parallel to the reward gradient, which again holds for any baseline of the weight update. The choice of baseline E again minimizes the update noise (SN 1, Eq. (S6)).

The NP update rule effectively incorporates an error backpropagation step, which reduces to a simple multiplication with r_{jt} due to the linearity of the spatial synaptic input summation. This allows to perturb only summed inputs instead of individual weights and may be expected to increase the performance of NP compared to WP (3, 12, 16, 18).

Theoretical analysis. We analytically compare WP and NP for temporally extended tasks by training a set of M linear perceptrons with N inputs. The task is to learn the mapping of a single fixed input sequence of duration T to a target output sequence in reward-based manner. This is motivated firstly by biological motor tasks that require such a mapping, like the learning of their single song in certain songbirds (Discussion). Secondly, it yields novel insights as it is the opposite extreme case to having no time dimension and different, random inputs in each trial; this case was treated analytically by (11) (Introduction). Thirdly, our findings yield an understanding of the learning performance for more general temporally extended tasks and networks studied later in this article. The analysis shows how learning depends on task dimensions and the structure of the input. Furthermore it reveals specific disadvantages of WP and NP. In Sec. "Multiple subtasks" we relax the assumption of exactly repeated inputs.

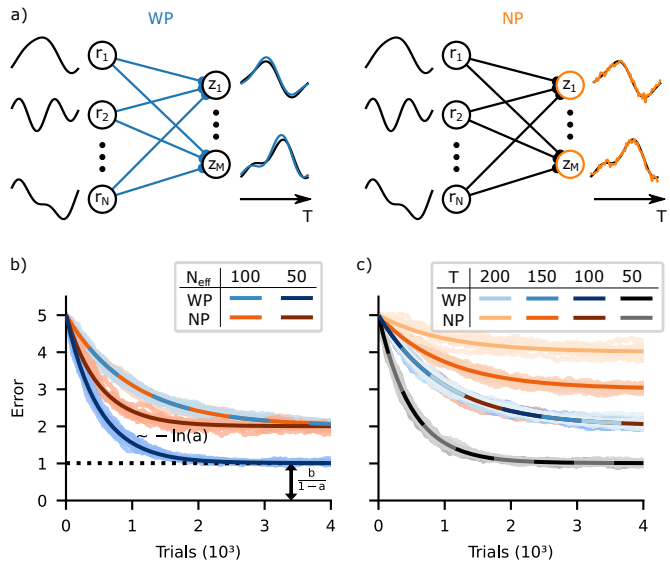

Fig. 1. Learning of temporally extended tasks in linear networks. a) Schematic setup of WP and NP. The M outputs z_i are weighted sums of the N inputs r_j. Left: WP perturbs the weights at the beginning of a trial; the resulting perturbations of the weighted sums of the inputs and thus the outputs reflect the dimensionality and smoothness of the inputs (blue). Right: NP perturbs the weighted sums of the inputs with dynamical noise (orange). b) WP (blue) works just as well or, in terms of the final error, better than NP (orange) when learning a single temporally extended input-output mapping. The error decay time decreases for WP and NP likewise with decreasing effective input dimension N_{eff} (light vs. dark curves). In contrast, the residual error only decreases for WP. c) Increased trial duration T does not change the progress of WP learning (blue curves lie on top of each other). In contrast, increasing T hinders NP learning by increasing the residual error (compare the increasingly lighter orange curves for larger T). If T decreases N_{eff} (gray curves), convergence is faster and to a lower residual error in both WP (because of the decrease in N_{eff}) and NP (because of the decrease in N_{eff} and T). Panel (b) shows error curves from simulations (10 runs, shaded) together with analytical curves for the decay of the expected error (solid), for fixed $T, N = 100$, $M = 10$ and $N_{\mathrm{eff}} \in \{100, 50\}$; the effective perturbation strength is $\sigma_{\mathrm{eff}} = 0.04$. Theoretical curves and simulations agree well. For WP and $N_{\mathrm{eff}} = 50$ the decay rate $(-\ln(a))$ and the residual error (dashed line) are highlighted. Panel (c) shows error curves from simulations and theory similar to (b) for fixed $N = 100$ and $T \in \{200, 150, 100, 50\}$. N_{eff} is set to 100 but cannot be greater than T, such that $T = 50$ forces $N_{\mathrm{eff}} = 50$.

The perceptrons generate as outputs the product of their $M \times N$ weight matrix w with the inputs,

$$z_{it} = \sum_{j=1}^{N} w_{ij} r_{jt}, \tag{7}$$

where $i = 1, \ldots, M$, Eq. (1) and Fig. 1a). For now we assume that the target output can be produced with target weights w^*, that is $z_{it}^* = \sum_{j=1}^{N} w_{ij}^* r_{jt}$. This condition will be alleviated in Sec. "Unrealizable targets". The learning goal is to reduce the quadratic deviation of each output from its target, which can be expressed through the weight mismatch $W = w - w^*$ and the

input correlation matrix $S_{jk} = \frac{1}{T}\sum_{t=1}^{T} r_{jt} r_{kt}$ (7),

$$E = \frac{1}{2T}\sum_{i=1}^{M}\sum_{t=1}^{T}(z_{it} - z_{it}^*)^2 = \frac{1}{2}\,\mathrm{tr}[WSW^T]. \qquad \textbf{(8)}$$

We note that with this quadratic error function the average weight update (cf. Eq. (4)) follows the gradient exactly, both for WP and NP (Eqs. (S16,S17)). We assume that the inputs are composed of N_{eff} orthogonal latent inputs. Since there are at most T linearly independent vectors of length T, the effective input dimension N_{eff} is bounded by $N_{\text{eff}} \leq T$. In biological systems, inputs are low dimensional; N_{eff} is often of the order of 10 (Discussion), in particular $N_{\text{eff}} \ll N$. As long as inputs are summed linearly, for clarity we will then hypothetically "rotate" the inputs such that only the first N_{eff} inputs are nonzero and equal to the latent ones (Fig. 2a). This does not affect the WP or NP learning process, because all perturbations are isotropic and the error function is rotationally invariant (SN 3). For clarity we further assume that all latent inputs have the same strength α^2, i.e. $\frac{1}{T}\sum_{t=1}^{N} r_{it}^2 = \alpha^2$ for the nonzero inputs $i = 1, ..., N_{\text{eff}}$.

Error dynamics. To elucidate the learning process and its dependence on the network and task parameters, we analytically derive the evolution of the expected error. This requires the computation of the error signal $E^{\text{pert}} - E$ and weight update after a given perturbation to determine the new error. Subsequent averaging over all perturbations yields the expected error at trial n, $\langle E(n)\rangle$, as a function of $\langle E(n-1)\rangle$, specifically a linear recurrence relation

$$\langle E(n)\rangle = a\,\langle E(n-1)\rangle + b \qquad \textbf{(9)}$$

(see SN 2 for a detailed derivation). The speed of learning is determined by the convergence factor a, while the per-update error increase b limits the final performance. Learning will stop at a finite error when an equilibrium between gradient-related improvement and reward noise induced deterioration is reached. The recurrence relation is solved by

$$\langle E(n)\rangle = \Big(\langle E(0)\rangle - E_f\Big)a^n + E_f. \qquad \textbf{(10)}$$

For $a < 1$, the average error $\langle E(n)\rangle$ converges exponentially at a rate $-\ln(a)$ towards a finite final (residual) error of $E_f = \frac{b}{1-a}$, as shown in Figure 1b). Usually in our settings a is sufficiently close to 1 to well approximate the convergence rate by $-\ln(a) \approx 1 - a$.

To understand how learning depends on the task parameters, we first consider the speed of learning. The determining convergence factor,

$$a = 1 - 2\eta\alpha^2 + \eta^2\alpha^4(MN_{\text{eff}} + 2) \qquad \textbf{(11)}$$

(SN 3), is affected by two opposing effects: On average, updates follow the gradient, thus reducing the error. This is reflected by a reduction of a by $-2\eta\alpha^2\,(+\eta^2\alpha^4)$, responsible for convergence. However, updates fluctuate, adding a diffusive part to the weight evolution which slows convergence down. Although these fluctuations, having zero mean, do not influence the expected error to linear order, they do so quadratically. Thus their contribution to a, $\eta^2\alpha^4(MN_{\text{eff}} + 1)$, is quadratic in the learning rate η. It is approximately proportional to MN_{eff}, the number of relevant weights that read out from nonzero inputs: fluctuations in each of these weights yield the same contribution - the exception being the twice as strong fluctuations along the single gradient-parallel direction, which (together with the quadratic effect of the mean update) cause the $+2$ in Eq. (11).

The fluctuations originate from a credit assignment problem: Only the perturbation parallel to the error gradient can be credited for causing the linear part of the error signal $E^{\text{pert}} - E$. WP has no way of directly solving the credit assignment problem of identifying this direction. Thus the perturbations of all MN weights are equally amplified in the constructions of their updates, Eq. (3), so that all weights fluctuate. This entails fluctuations in the MN_{eff} relevant weights, which influence output and error. NP can at least partially solve the credit assignment problem by using eligibility traces, which are zero for weights that read out from zero inputs. By projecting each of its (M) T-dimensional output perturbations onto the effectively N_{eff}-dimensional inputs, NP restricts its updates to the MN_{eff}-dimensional subspace of relevant weights. The convergence speed thus becomes independent of T as for WP. Interestingly, WP and NP therefore converge at the same speed despite their different numbers of fluctuating weights. The reason is that the fluctuations of the relevant weights are the same for both algorithms.

The balance between the improvement resulting from following the gradient ($\sim \eta$) and the deterioration due to the fluctuations of task-relevant weights ($\sim \eta^2$) in Eq. (11) is controlled by the learning rate: small learning rates imply averaging out fluctuations over many updates and therefore dominance of gradient following, leading to convergence. For the remainder of the analysis of this setting, both algorithms will be compared at their optimal learning rate η^*, which is defined to yield fastest convergence, in other words: to minimize a. This definition is chosen because it is conceptually straightforward and Eq. (11) directly leads to the simple expressions

$$\eta^* = \frac{1}{(MN_{\text{eff}} + 2)\alpha^2}, \qquad a^* = 1 - \frac{1}{MN_{\text{eff}} + 2}. \qquad \textbf{(12)}$$

Here, the factor $1/\alpha^2$ in η^* cancels the scaling of the gradient with the input strength and equals the optimal learning rate for GD (Eq. (S28)). In order to allow for averaging out the update fluctuations, WP and NP learning additionally have to slow down by a factor of approximately MN_{eff}. Learning diverges for $\eta \to 2\eta^*$ where $a \to 1$. Eq. (12) shows that WP's convergence rate is worse than GD's by a factor generally smaller than the number of weights. Further, NP's convergence rate is worse by a factor generally larger than the number of nodes. Thus, the number of weights or nodes is insufficient to predict the performance of WP or NP, respectively.

The per-update error increase and the final error, b and E_f, result from finite perturbation sizes. Finite perturbation sizes lead, due to the curved, quadratic error function, to an esti-

mate that is at least slightly incompatible with the linear approximation assumed by the update rules (cf. Eq. (4)). This is particularly apparent when the output error and thus the gradient is (practically) zero: Any finite weight or node perturbation then leads to an increase of the error and thus to an opposing weight update instead of no weight modification. This prevents the weights from reaching optimal values and results in a finite final error E_f. The described difference between perturbation-based error estimate and linear approximation is a form of "reward noise". It is nonzero only for finite perturbation size, as reflected by the dependence of b and E_f on σ (which is quadratic due to the quadratic error nonlinearity).

For a fair comparison of WP and NP we choose σ_{WP} and σ_{NP} such that they lead to the same effective perturbation strength σ^2_{eff}, as measured by the total induced output variance. This leads to $\sigma^2_{\mathrm{NP}} = \sigma^2_{\mathrm{eff}}$ and $\sigma^2_{\mathrm{WP}} = \frac{1}{\alpha^2 N_{\mathrm{eff}}} \cdot \sigma^2_{\mathrm{eff}}$ (SN 1). Evaluated at the optimal learning rate η^*, the leading order term of the final error is

$$E_f^{\mathrm{WP}} = \frac{b^*_{\mathrm{WP}}(\eta^*)}{1-a^*} \approx \tfrac{1}{8}\sigma^2_{\mathrm{eff}} \cdot M^2 N_{\mathrm{eff}}, \tag{13}$$

$$E_f^{\mathrm{NP}} = \frac{b^*_{\mathrm{NP}}(\eta^*)}{1-a^*} \approx \tfrac{1}{8}\sigma^2_{\mathrm{eff}} \cdot M^2 T. \tag{14}$$

Importantly, the final error of WP is here generally smaller, by a factor $N_{\mathrm{eff}}/T \leq 1$. To understand this, we focus for both WP and NP on the output perturbations that they generate. By perturbing the weights, WP induces output perturbations that are linear combinations of the inputs. These are confined to the effectively MN_{eff}-dimensional subspace in which also the (realizable part of the) output error gradient $(z - z^*)/T$ lies. NP, on the other hand, creates an entirely random MT-dimensional perturbation vector (Fig. 1a). Only the projection of this vector onto the output gradient is useful for learning. This projection is smaller for NP's random vector, since the vector has effectively a larger dimensionality than the output perturbation vector of WP, at the same length. NP compensates this deficit by amplifying the smaller gradient projection more strongly. It thus achieves the same mean update and convergence speed as WP. However, it also more strongly amplifies the reward noise that comes with larger perturbation sizes, which results in a larger final error. The scaling of E_f with $M^2 N_{\mathrm{eff}}$ or $M^2 T$ reflects the effective output perturbation dimensions, MN_{eff} or MT of WP or NP, and additionally the general scaling of errors with M (Eq. (8)) (SN 3, Eqs. (S120,S121)).

Taken together, we observe that here WP learning works just as well as or better than NP. Both algorithms have the same speed of convergence, but the final error E_f for WP is smaller or equal compared to NP. The rate of convergence decreases with increasing M and N_{eff}. Longer trial durations T harm NP by linearly increasing E_f. Larger effective input dimensionality N_{eff} similarly harms WP.

Weight diffusion. When the input has less than maximal dimensionality, $N_{\mathrm{eff}} < N$, only certain combinations of weights read out nonzero components of the input. This becomes particularly clear for the considered rotated inputs: If WP adds a perturbation to a weight mediating zero input, to an irrelevant weight, the output and the error remain unchanged. This missing feedback leads to an unbounded diffusion-like random walk of irrelevant weights. For unrotated inputs, the weight strength diffuses in irrelevant weight space directions. We will see below (sec. "Multiple input patterns") that the weight diffusion harms performance when learning multiple input-output patterns.

We find that for WP in the limit of infinitesimally small perturbations $\sigma_{\mathrm{WP}} \to 0$, all weights initially change and then converge (SN, Fig. S1). This is because the learning-induced drift of relevant and the diffusion of irrelevant weights both stop when the error converges to zero: the error E is quadratic, such that for infinitesimally small perturbations $E^{\mathrm{pert}} = E$ at its minimum. In contrast, for finite perturbations a residual error remains and weights continue to change. In particular, irrelevant weights continue to diffuse, Fig. 2b.i. Standard mechanisms such as an exponential weight decay (28, 29) confine their growth, but also bias the relevant weights towards zero and therewith increase the residual error, Fig. 2b.ii. The quantitative details of the weight diffusion process can be analytically understood (SN 4).

NP does not generate weight diffusion: the rotated inputs render it obvious that in NP the eligibility trace (Eq. (6)) selects only the weights from relevant inputs to be updated, since for irrelevant inputs we have $r_{jt} = 0$ for all t such that $\Delta w^{\mathrm{NP}}_{ij} = 0$. This qualitative difference indicates that weight diffusion may be experimentally measured to distinguish whether WP or NP underlie learning of a certain task. In fact, random walk-like weight changes that are unrelated to neuronal activity and task learning are common in biological neural networks (30).

Unrealizable targets. In the previous sections we assumed that the target outputs z^*_{it} could be exactly realized by setting the perceptron weights w equal to some target weights w^*, $z^*_{it} = \sum_{j=1}^N w^*_{ij} r_{jt}$. In general, however, the target outputs may contain components d that cannot be generated by the network, which is limited to producing linear combinations of the inputs. Unrealizable components are orthogonal to all inputs when interpreted as T-dimensional vectors, $\sum_{t=1}^T d_{it} r_{jt} = 0 \; \forall i, j$. The target may be written as a sum of realizable and orthogonal unrealizable parts, $z^*_{it} = \sum_{j=1}^N w^*_{ij} r_{jt} + d_{it}$. An illustration of such a target is given in Fig. 3a.

WP induces output perturbations $\delta z_{it} = \sum_{j=1}^N \xi^{\mathrm{WP}}_{ij} r_{jt}$, which are linear combinations of the inputs. The components of $z^{\mathrm{pert}} - z^*$ that are orthogonal to all inputs, d, thus always remain unchanged, irrespective of the current student weights and applied perturbations. This leads to the same constant additive contribution $E_{\mathrm{opt}} = \frac{1}{2T}\,\mathrm{tr}[dd^T]$ to the perturbed and unperturbed errors E^{pert} and E (Eq. (8)). It cancels in the weight update rule (Eq. (3)) such that WP learning is unchanged and Eq. (10) still holds when shifting its final error to $E^{\mathrm{WP}}_{f,\mathrm{unr}} = E^{\mathrm{WP}}_f + E_{\mathrm{opt}}$ (Eq. (S119)). E_{opt} marks the minimum error that necessarily remains even with $w = w^*$, due to the unrealizable components.

In contrast, NP perturbs the outputs with white noise. This noise generally has a nonzero component along d, which affects E_{pert}. Since such a component cannot be realized through an update of the weights, the resulting change of the

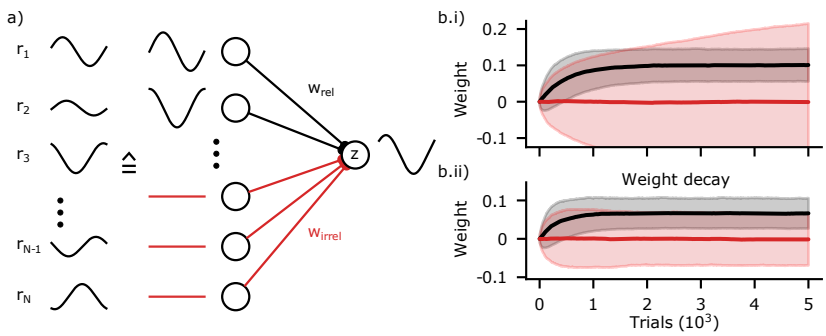

Fig. 2. Hypothetical rotation of inputs and weight diffusion. a) Because the inputs (left, black) are summed linearly, they can be "rotated" so that for our tasks the first N_{eff} inputs are nonzero and agree with the latent inputs (middle, black). The remaining inputs are then zero (middle, red) and their weights are irrelevant for the output (right, red). b.i) In WP with finite perturbation size σ_{WP}, the irrelevant weights diffuse without bounds (red), while the relevant weights converge and fluctuate (black) around the teacher weights. Displayed are the mean (solid) and standard deviation (shaded area) of the weight ensembles. b.ii) Weight decay confines the diffusion. In (a) $N_{\text{eff}} = 2$, the latent inputs are a sine and a cosine. Parameters in (b.i,ii): $M = 10$, $N, T = 100$, $N_{\text{eff}} = 50$, $\sigma_{\text{eff}} = 0.04$, teacher weights $w_{\text{rel},i} = 0.1$, weight decay $\gamma_{\text{WD}} = 0.999$; results are averaged over 10 runs.

error is non-instructive and represents reward noise that adds noise to the updates. Consequently, while the convergence factor a remains unchanged, the final error of NP increases more strongly than for WP, Fig. 3b. At the optimal learning rate the increase in final error due to unrealizable target components is twice that of WP, $E^{\text{NP}}_{f,\text{unr}} \approx E^{\text{NP}}_f + 2E_{\text{opt}}$ (SN 3, Eq. (S123)). For $E_{\text{opt}} > E^{\text{NP}}_f$ the coupling of node perturbations to unrealizable target components becomes NP's main contribution to its final error and should not be neglected.

Multiple subtasks. In general learning tasks, inputs and targets may vary from trial to trial. To obtain an intuition for how this affects the speed of WP and NP learning, we here consider a simplified case: The goal is to solve a task with an overall effective input dimension of $N^{\text{task}}_{\text{eff}}$. The task has the same properties as the tasks before when each trial was identical. In particular, it has $N^{\text{task}}_{\text{eff}}$ orthogonal latent inputs of strength α^2 and the inputs are rotated such that only the first $N^{\text{task}}_{\text{eff}}$ inputs are nonzero. The task is, however, not presented as a whole, but in pieces: in each trial a random subset of $N^{\text{trial}}_{\text{eff}}$ out of the first $N^{\text{task}}_{\text{eff}}$ inputs are set active to train the network. The error in an individual trial then only depends on its $MN^{\text{trial}}_{\text{eff}}$ trial-relevant weights, while the performance on the full task depends on the $MN^{\text{task}}_{\text{eff}}$ task-relevant weights. The ratio $N^{\text{task}}_{\text{eff}}/N^{\text{trial}}_{\text{eff}} = P$ marks the number of trials needed to gather information on all task-relevant weights.

NP only updates the weights relevant in a trial (Sec. "Weight diffusion"). Also for tasks consisting of multiple subtasks it can thus operate at the learning rate that is optimal for a trial, $\eta^*_{\text{NP}} = 1/(MN^{\text{trial}}_{\text{eff}} + 2)\alpha^2$ (cf. Eq. (12)). Because an update only improves $MN^{\text{trial}}_{\text{eff}}$ of the $MN^{\text{task}}_{\text{eff}}$ task-relevant weights, the convergence rate $-\ln a \approx 1 - a$ of the expected error, averaged over the input distribution, is smaller by a factor of $1/P$ than for a single input pattern,

$$a^*_{\text{NP}} = 1 - \frac{1}{P}\frac{1}{MN^{\text{trial}}_{\text{eff}} + 2}. \tag{15}$$

WP, on the other hand, updates all weights such that the weights that are irrelevant for the trial are changed randomly (Sec. "Weight diffusion"). This worsens the performance for the inputs of other trials. Because there are now $MN^{\text{task}}_{\text{eff}}$ task-relevant weights whose fluctuations hinder learning, WP has an optimal learning rate of only $\eta^*_{\text{WP}} = 1/(MN^{\text{task}}_{\text{eff}} + 2)\alpha^2$ (SN 5). As for NP, each trial's progress is only on $1/P$ of the task-relevant weights, so that the optimal convergence factor for WP on the full task is

$$a^*_{\text{WP}} = 1 - \frac{1}{P}\frac{1}{MN^{\text{task}}_{\text{eff}} + 2}. \tag{16}$$

The convergence of WP is slower than that of NP by roughly $1/P$, the ratio of $N^{\text{task}}_{\text{eff}}$ and $N^{\text{trial}}_{\text{eff}}$, Fig. 4c.

Our results have concrete implications for learning of multiple actions such as sequences of movements (31). They can be learned by splitting them into subsets, which are called (mini-)batches in machine learning. In our terminology, each batch corresponds to a subtask, the number of batches to P, the dimensionality of the input data to $N^{\text{task}}_{\text{eff}}$ and the batch size N_{batch} to $N^{\text{trial}}_{\text{eff}}$, assuming for simplicity that individual data points are pairwise orthogonal and have no time dimension. For $MN^{\text{trial}}_{\text{eff}} \gg 2$, Eqs. (15,16) thus imply that the convergence rate of NP is independent of the batch size while that of WP is proportional to the batch size and reaches NP's convergence rate for full batch learning (Fig. S3). The same holds for the optimal learning rates as α^2 scales inversely with the batch size.

Conclusions from the theoretical analysis and new learning rules. Our theoretical analysis reveals a simple reason for the differences between WP and NP: WP produces better perturbations, while NP better solves the credit assignment problem. Output perturbations caused by WP lie, in contrast to NP, always in the realizable output subspace and do not interfere with unrealizable target components. On the other hand, NP updates only (trial-)relevant weights, while WP updates all weights so that (trial-)irrelevant weights change randomly. When single trials capture only a small part of the full task, this slows down WP learning. Training in batches reduces the disadvantage.

Based on these insights, we introduce two novel learning rules, WP0 and Hybrid Perturbation (HP), Fig. 4a,b. WP0 adds a simple modification to WP: not to update currently irrelevant weights, i.e. weights whose inputs are zero (or close to it). This solves part of WP's credit assignment problem, as changing the weights will not improve the output, and it avoids diffusion of irrelevant weights. The improvement is especially large when inputs are sparse so that many inputs are (close to) zero (Fig. 4), which might be frequently the case in biological neural networks (32–34). HP aims to combine the advantages of WP and NP by generating the output perturbations like WP, through perturbing the weights, and generating updates like

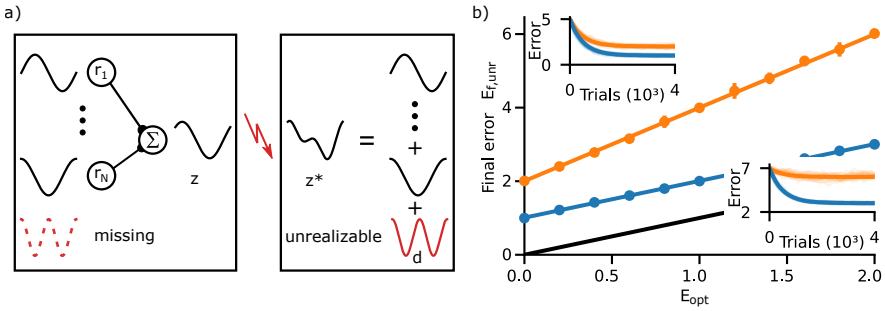

Fig. 3. Unrealizable target components harm NP learning. a) General targets may contain a component that is perpendicular to any input and thus unrealizable (red). b) Final error after convergence as a function of the error E_{opt} that necessarily remains, since the target is unrealizable. The final error of WP (blue) is only shifted by E_{opt}, that of NP (orange) increases twice as fast, by approximately $2E_{\mathrm{opt}}$. Data points: mean and standard deviation (averaged over 10 simulated runs) of the final error. Curves: theoretical predictions. Black: E_{opt}. Insets: error dynamics for $E_{\mathrm{opt}} = 0$ (left) and $E_{\mathrm{opt}} = 2$ (right). Parameters: $M = 10$, $N, T = 100$, $N_{\mathrm{eff}} = 50$, $\sigma_{\mathrm{eff}} = 0.04$.

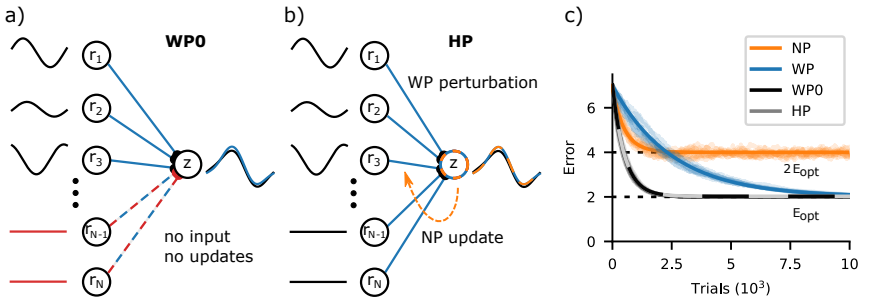

Fig. 4. New learning rules and learning of tasks consisting of multiple subtasks. a) WP0 does not update weights that mediate zero input, avoiding their diffusion. b) Hybrid perturbation (HP): NP scheme with output perturbations induced by WP. c) WP converges approximately $P = 5$ (number of subtasks) times slower than NP, but in presence of unrealizable target components (or for finite σ_{eff} and $N_{\mathrm{eff}}^{\mathrm{trial}} < T$, SN 5) to a lower final error. For the used maximally sparse and equally strong inputs, WP0 and HP combine the higher convergence rate of NP with the low final error of WP. Error curves (solid: theoretical predictions, shaded: 10 exemplary runs) are for $M = 10$, $N, T = 100$, $N_{\mathrm{eff}}^{\mathrm{task}} = 50$, $N_{\mathrm{eff}}^{\mathrm{trial}} = 10$, negligible σ_{eff} and $E_{\mathrm{opt}} = 2$.

NP, using its eligibility trace. The learning rule performs well when all latent inputs have (approximately) the same strength α^2, Fig. 4b,c.

WP0 and HP perform for the tasks used in the theoretical analysis section as well as the better of WP and NP, or better than both (Fig. Fig. 4c). WP0 is, however, benefited by the assumption of rotated inputs (in contrast to WP and NP), as it renders the input maximally sparse. Further, the latent inputs have equal strengths, benefiting HP. We observe only slight improvements of WP0 over WP for the reservoir computing and MNIST task, due to the lack of coding sparseness in our networks. HP performed much worse than WP and NP in the reservoir computing and similar to NP in the MNIST task. We explain this by the relevance of weak inputs (SN7). Adding appropriately equalizing preprocessing layers may mitigate HP's problems. Further, weak inputs may be irrelevant for biological learning.

Simulated learning experiments. In the following, we apply WP and NP to more general networks and temporally extended tasks with nonlinearities. We cover reservoir computing for dynamical pattern generation, learning of recurrent weights in a delayed non-match-to-sample task and a temporally extended, reward-based learning version of MNIST. The results confirm and extend our findings for analytically tractable tasks: they often show similar and superior performance of WP in temporally extended tasks relevant for biology and machine learning.

Reservoir computing-based drawing task. In reservoir computing schemes an often low-dimensional input is given to a recurrent nonlinear network. The network effectively acts as a nonlinear filter bench: it expands the input and its recent past by applying a set of nonlinear functions to them. Each unit outputs one such function, which depends on the interactions within the recurrent network. Like a "computational reservoir", the network thereby provides in its current state the results of manifold nonlinear computations on the current and past inputs. A desired result can be extracted by training a simple, often linear readout of the reservoir neurons' activities. Reservoir computing schemes are widely used as models of neurobiological computations (35–39), since learning in them is simpler and seems more easily achievable with biological machinery than learning of full recurrent and multilayer networks. Further, the schemes explain the presence of inhomogeneity and apparent randomness in neuron properties and connectivity in biological neural networks as helpful for enriching the computational reservoir. Here we find that when learning temporally extended output patterns with a reservoir computing scheme, WP can learn as well as or better than NP.

We consider a recurrently connected reservoir of $N = 500$ rate neurons driven by five external inputs of length $T = 500$. Inspired by the behaviorally relevant task of reproducing a movement from memory - here drawing a figure - the task is to generate the x and y coordinates of a butterfly trajectory (40, 41) at the $M = 2$ outputs by training a linear readout (Fig. 5a). The trajectory is non-trivial in that it is not realizable from the external inputs, but requires reading out from many reservoir modes (Fig. 5b, dashed gray line).

Formally, the task is similar to the setting discussed above, with the difference that there is a wide distribution of different, nonzero input strengths α_μ^2. The evolution of expected error is then best described by splitting the error $E = \sum_{\mu=1}^{N} E^\mu$ into different error components, each of which is associated with the weights that read out from a latent input $r_{\mu t}$ (SN 6). Their evolution follows a matrix exponential where different components decay at different rates and interfere with each other. Components that decay relatively quickly may be the main source of improvements in the beginning of training, whereas more slowly decaying components dominate the error towards the end. This effect can be seen in the approximately piecewise linear error decay in the logarithmic plot in Fig. 5c.

Fig. 5c compares the performance of WP and NP in the drawing task. Perturbation size is finite, $\sigma_{\text{eff}} = 5 \times 10^{-3}$. WP converges faster initially, which may be typical for tasks with distributed input strengths (SN 6). It also achieves a lower final error. This is compatible with the observation that the effective dimension of the reservoir dynamics, as measured by the participation ratio ($\text{PR} \approx 5$), is much smaller than the temporal extent of the task: the resulting smaller effective perturbation dimension of WP ($M\text{PR}$ vs. MT for NP) yields an advantage for WP (Fig. 1b). For the biologically less relevant case of infinitesimal perturbation sizes the performances of WP and NP are similar, Fig. S4 (compatible with Fig. 1 with $b = 0$). In both cases, towards larger trial numbers the convergence of WP becomes slower: WP has difficulties with adjusting weights mediating weak inputs, since the impact of their perturbation on the error is small; the same effect underlies the weight diffusion in Fig. 2. Simulations indicate that the convergence is only slower by a constant factor of the order of 1 and that the optimal learning rate can be well estimated from the participation ratio (Fig. S5).

Delayed non-match-to-sample task. To ensure analytical tractability and for simplicity, so far we made a few biologically implausible assumptions. Specifically, only connection weights to linear units were trained, each trial consisted of a perturbed and an unperturbed run and mostly the exact same input was used in each trial. In the following we show that our findings generalize to settings without these assumptions. For this we consider the learning of a delayed non-match-to-sample (DNMS) task (temporal XOR) by nonlinear recurrent networks. DNMS tasks and closely related variants are widely used both in experiment (42) and theory (16, 43), where they serve as simple nonlinear, working memory-reliant decision making tasks. We use the same setting as (16), which shows that a new variant of NP is able to solve the DNMS task. In particular, the setting is not adjusted to WP. The network consists of 200 nonlinear rate neurons receiving input from two external units u_1 and u_2. One of the network neurons, whose rate we denote with z, serves as its output (Figure 6a). In each trial, the network receives two input pulses, where each pulse is a $200\,\text{ms}$ long period with either u_1 or u_2 set to 1, and subsequently has to output 1 for $200\,\text{ms}$ if different inputs were presented and -1 if the same inputs were presented (Figure 6b). There is a $200\,\text{ms}$ long delay period after each input pulse.

We train all recurrent weights using the usual update rules (Eqs. (3,6)), but replace the error of the unperturbed trial by an exponential average of the errors of the previous trials (14–16). Hence, each trial now only consists of a perturbed and not additionally an unperturbed run. We first assume that the exact perturbations ξ are accessible for the weight update, which seems biologically plausible for WP (cf. Discussion), but less so for NP (cf. Discussion and (16)). Therefore we also compare WP and NP to the biologically plausible version of NP proposed by (16), which avoids this assumption: in the weight update rule, it approximates the exact node perturbations ξ^{NP} with a nonlinearly modulated difference between the momentary input to a neuron and its short term temporal average (see Methods for more details).

Figure 6c shows the performance of the three update rules in terms of their accuracy over the last 100 trials, where a trial is considered successful if the mean absolute difference between z and the target output is smaller than 1. We find that all update rules learn the task comparably well and reach perfect accuracy within at most 2000 trials when considering the median of network instances. Thus, our previous findings that WP can perform as well as or better than NP in simplified settings extend to the considered biologically plausible setup. That means WP can perform well for nonlinear neuron models, recurrent connectivity and when the error of the unperturbed network is not available. Furthermore, the results indicate that approximating the perturbation as in (16) only mildly impacts the performance of NP for the considered task.

MNIST. Finally, we apply WP and NP to MNIST classification. We use batches of images to train the networks. Each time step thereby corresponds to the presentation of one image and the networks receive error feedback only at the end of a batch. This allows us to test how well WP and NP work on a more complicated, temporally extended task and in networks with a multi-layer structure. In addition, it allows us to study how our analytical results for the learning of multiple input patterns (sec. "Multiple subtasks") extend to real-world tasks.

We use a two-layer feed-forward network with 10 output neurons and 100 neurons in the hidden layer (Fig. 7a). It learns via the rules Eqs. (3,6), where T equals the batch size N_{batch}. Hence, the perturbation is different for each image in the case of NP, while it is the same for WP. We test WP and NP for batch sizes of $N_{\text{batch}} \in \{1, 10, 100, 1000\}$. For each batch size we determine the best-performing learning rates η and perturbation strengths σ^2_{WP}, σ^2_{NP} via grid searches. The perturbation strength has, however, little impact on performance, indicating that the final error is not restricted by reward noise due to finite size perturbations (Eqs. (13,14)).

We find that for WP the performance improves drastically with increasing batch size, Fig. 7b. The final test accuracy is only about $69\,\%$ for a batch size of 1 but reaches $92\,\%$ for $N_{\text{batch}} = 1000$. Simultaneously the optimal learning rate increases strongly, by a factor of approximately 50 (Fig. S6c and Supplementary Table S2). For comparison: the biologically implausible stochastic gradient descent (SGD) rule reaches accuracies of $95\,\%$–$98\,\%$ for the considered batch sizes. In contrast, the learning curves of NP appear to be entirely independent of the batch size (Figure 7b); the final test accuracy is always about $86\,\%$ and the optimal learning rate is constant as well. Hence, larger batch sizes, as commonly used in machine learning, favor WP, while smaller batch sizes favor NP.

An improvement of WP with batch size and NP's independence of it are in agreement with our theoretical analysis Sec. "Multiple inputs". However, from this analysis we also expected that WP's learning rate can reach at most that of NP for large batch size. NP's slower convergence suggests that it is more susceptible to deviations of the network architecture from linear, single-layer networks. Indeed, when using single-layer networks, NP's performance improves, while the opposite holds for WP and SGD (Fig. S7). In a single layer linear network with realizable targets the prediction of our analysis is recov-

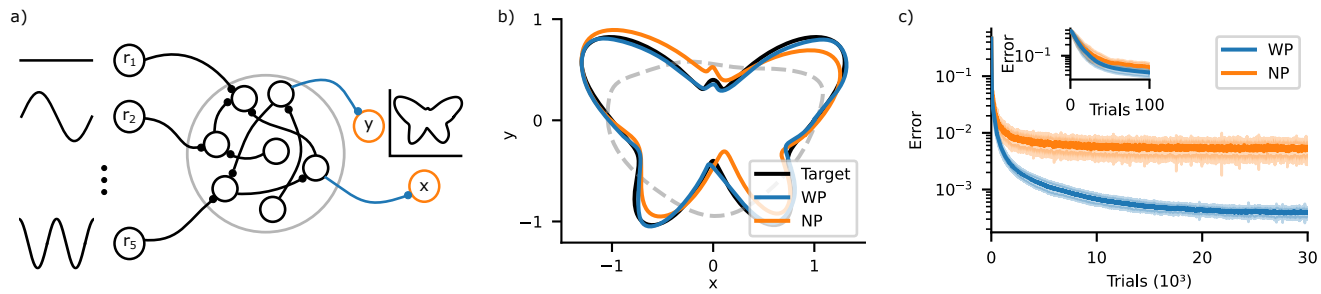

Fig. 5. WP outperforms NP on a reservoir computing-based drawing task. a) Schematic of the recurrent, fixed reservoir receiving five external inputs. Only readout weights are learned. b) Target (black) and final outputs of WP (blue) and NP (orange). A least squares fit (gray, dashed) using only the first five principle components of the reservoir dynamics demonstrates that the task critically depends on reading out further, weaker dynamical components. c) Error curves on a logarithmic scale. WP reaches a lower final error, cf. also (b). The curves show mean and standard deviation over 100 runs of the same network with different noise configurations. Inset: Early error evolution. There is a considerable improvement already during the first 50 trials.

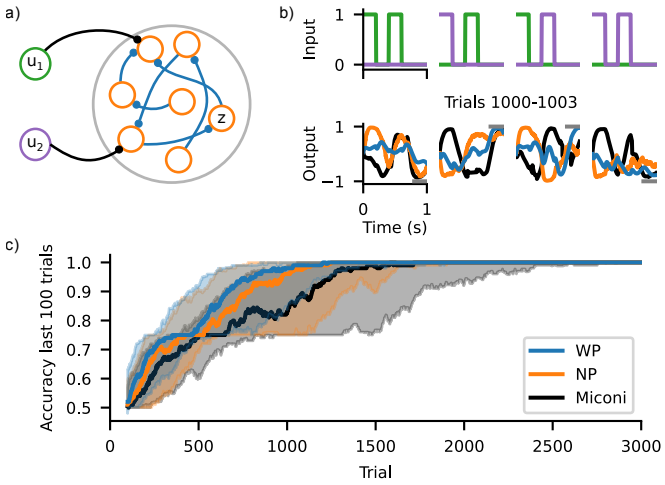

Fig. 6. WP performs as well as NP on a DNMS task. a) Schematic of the recurrent network with inputs u_1 and u_2 and output z. All network weights are learned, i.e., for WP, all network weights (blue) are perturbed and for NP, all network nodes (orange) are perturbed. b) Inputs and outputs during example trials. Top row: Inputs u_1 (green) and u_2 (purple) for the four different trial types. Bottom row: Outputs for WP (blue), NP (orange) and the version of NP proposed by (16) (black) for trials 1000–1003 for the inputs shown above. Gray bars show target outputs. c) Accuracy during training. WP (blue) performs similarly well as NP (orange) and the version of NP used by (16) (black). There is a noticeable transient slowdown at an accuracy of $75\,\%$, which corresponds to the successful learning of three out of the four different trial types. Solid lines show the median and shaded areas represent the interquartile range between first and third quartile using 100 network instances.

ered: NP performs better than WP even for large batch sizes (Fig. S7).

The results are particularly remarkable when naively comparing the number of perturbed nodes and weights ("Introduction"): For the network considered here, there are only 110 output and hidden nodes, but $79\,510$ weights (including biases). Nevertheless WP can clearly outperform NP. Also a comparison of the actual perturbation dimensions cannot explain the better performance of WP in, e.g., Fig. 7b lower left (WP pert. dim.: $79\,510$, NP pert. dim.: $110 \times T = 11\,000$).

Discussion

Our results show that WP performs better than NP for tasks where long trials capture most of the task's content. This might seem paradoxical as NP incorporates more structural knowledge on the network, namely the linear summation of inputs. However, WP accounts for the fact that the weights in a neural network are (approximately) static. Further, by perturbing the weights it implicitly accounts for low input dimensionality and generates only realizable output changes. Therefore it generates better tentative perturbations. This leads to less noise in the reward signal and better performance (smaller final error and sometimes faster convergence) in the tasks where WP is superior to NP.

Our theoretical analysis shows that the lower noise in WP firstly results from an effective perturbation dimension that is lower than NP's if the temporal extent of a task is larger than its input dimensionality, $T > N_{\text{eff}}$. Secondly, factors such as the attempt of NP to realize unrealizable targets contribute. Temporally extended tasks with durations on the order of seconds and low dimensionality occur frequently in biology, for example in motor learning and working memory tasks. In line with perturbation-based learning, biological movements are endowed with noise, which helps their learning and refinement (5). The associated neuronal dynamics in the brain are confined to a low-dimensional space, a property shared by many types of biological and artificial neural network activity (44–46). The dynamics for simple movements as investigated in typical experiments are embedded in spaces of dimension of order 10 (20). This indicates low effective input dimensionality N_{eff} at the different processing stages. The effective muscle activation dimensionality is similarly low (21, 47). Neurons under in vivo conditions can faithfully follow input fluctuations on a timescale of 10ms (48) and significant changes in neuronal trajectories happen on a timescale of 100ms (20, 22, 49). For the learning of a movement of duration 1s, this suggests a number of time bins T of about 10 to 100, similar to the expected input dimension. This implies that both WP and NP are promising candidates for the learning of simple movements. Our results indicate that WP will be superior if the movements are longer lasting.

We have explicitly studied the learning of movement generation (drawing task) and of a working memory task (DNMS). The numerical simulations show that WP performs similarly well or better compared to NP. In a task generally investigating the learning of complicated nonlinear, temporally extended input-output tasks (MNIST), WP outperforms NP as soon as the tasks have sufficient temporal extent.

As another concrete application, consider the learning of the single song in certain birds. A single, stereotypical input se-

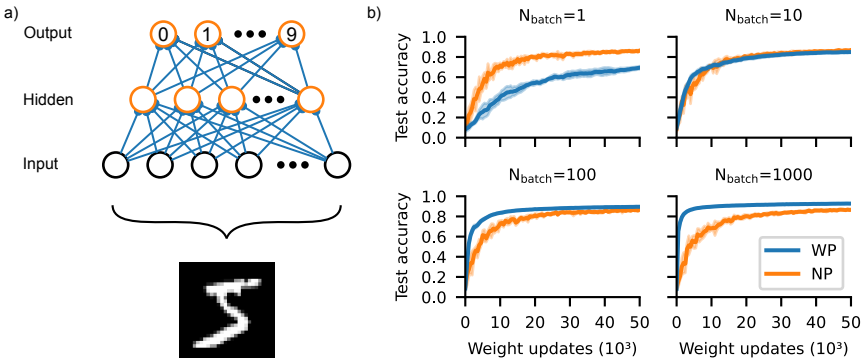

Fig. 7. WP can outperform NP on MNIST. a) Schematic of the used fully connected, two-layer network. All network weights are learned, i.e., for WP all network weights (blue) are perturbed and for NP all network nodes (orange) are perturbed. b) Test accuracy as a function of the number of weight updates for WP (blue) and NP (orange) for different batch sizes. NP does not profit from increasing the batch size and always reaches a final accuracy of $\approx 86\,\%$. WP improves considerably with increasing batch sizes and reaches a final accuracy of $\approx 92\,\%$ for $N_{\mathrm{batch}} = 1000$. Solid lines show the mean and shaded areas show the standard deviation using 5 network instances.

quence in a "conductor area" (HVC) may drive the circuit (33, 50). The effective input dimension N_{eff} is thus at most as large as the temporal dimension T of the task. Based on recent experiments, (50) proposed that the output of the tutor/experimenter area (LMAN) is modified by reinforcement learning via NP, such that it guides the motor area (RA) to learn the right dynamics. Our analytical results predict that WP is as well or better suited to achieve this task since $N_{\mathrm{eff}} \leq T$. Earlier work suggested that WP (23) or NP (18) may directly mediate the learning of the connections from HVC to RA. Due to HVC's very sparse activity, WP0 would be highly suitable for such learning. Reward-based learning of mappings between conductor sequences and downstream neural networks may also be important for different kinds of precisely timed motor activity (51, 52) and for sequential memory (53, 54).

WP and NP have biologically plausible implementations. NP requires that the plastic synapses can keep track of their input and the somatic perturbations (which may arrive from a tutor/experimenter neuron). Biologically plausible mechanisms for this have been proposed both for tasks with immediate reward (14, 15) and reward at a temporally extended trial's end (16). Their underlying idea is to assume that the perturbation fluctuates more quickly than the other input. The present fluctuation can then be approximately isolated by subtracting a short term temporal average of the past overall input from the present one (14, 15). This difference replaces the injected perturbation in the eligibility trace. For tasks with late reward, the eligibility trace needs to integrate a nonlinearly modulated version of the described difference (16). This prevents the cancellation of a perturbation's effect by the subsequent change in the average that it evokes, because the peak in the original perturbation is sharper and higher than the one in the average. We use this learning model of (16) in Figure 6. The biological implementation of WP may be even simpler. A neural network needs to generate labile random weight changes and keep track of them. They should be approximately constant during a task and enhanced, deleted or reversed by a subsequent reward signal. Experiments on timescales from minutes to days found spontaneous changes in the synaptic weights, which have similar strength as changes due to activity dependent plasticity (30). Such changes might generate the perturbations required for our WP scheme. Previous work suggested also synaptic unreliability to provide the perturbations for WP (55). This fits into our scheme of static weight perturbations if neurons spike once during a trial or if they burst once and the synaptic transmission is restricted to a single time bin. Another source of the required randomness may be fluctuations of activity-dependent plasticity, while the deterministic baseline acts as a useful prior. If the baseline is unrelated to the task, it will with high probability be orthogonal to task-relevant directions (due to the high dimensional weight space) and not harm learning, similar to the weight diffusion in WP. In this way, the fluctuations of activity-dependent plasticity, rather than their deterministic part, may be the source of learning.

Modulation of weight changes by reward has been observed in various experiments (56, 57). As an example, the potentiation of synapses is enhanced or reversed depending on the presence or absence of a temporally close dopamine reward signal (58). Also other factors play a role; potentiation can for example be reversed within a "grace period" of tens of minutes by a change of environment (59). The consolidation and amplification of changes may be dependent on plasticity related proteins, which are upregulated by reward and for which the synapses compete (synaptic tagging hypothesis) (57, 60). A posteriori modifications of tentative synaptic weight changes are also assumed in the reinforcement learning scheme of reward modulated Hebbian plasticity (61, 62), which is closely related to WP.

WP has been proposed in several variants. They differ in: (i) the task setup, for example instantaneous (9, 11) or temporally extended tasks (10, 23, 55), (ii) the implementing network, for example rate (23) or spike-based (55, 63) networks, (iii) the perturbation scheme, where all weights are simultaneously perturbed (9, 10, 55) or only one weight at a time (17), (iv) the computation of the weight update, by correlating reward and perturbation (9–11, 13, 55) or direct estimation of the gradient components (for the single weight perturbation scheme) (11, 17), (v) the estimation of the success of the perturbed network, which may involve a comparison of the obtained reward to an unperturbed baseline (10, 11) or a running average (23, 55) or it may consider the reward only (9, 55) and (vi) the weight update, which may be proportional to the success of the network (9–11, 13, 55) or independent of its size as long as there is an improvement (23). A similar diversity of NP variants exists (11, 14–16, 19, 25, 64).

The tasks considered in our article are temporally extended. The reward is provided at the end of the trial, but influenced by earlier output states. This is consistent with many tasks in biology (1, 7, 16, 23) and with the learning schemes by

(10, 12, 16, 23). We choose a WP rule that is biologically plausible as it involves simultaneous perturbations to all weights and correlates reward and weight change. The success measure compares the obtained reward to the reward of an unperturbed network in order to reduce the update noise (10, 11). In particular this avoids that unfavorable perturbations are associated with positive reward feedback. Finally, the weight update is proportional to the measured success in order to ensure that it occurs on average parallel to the reward gradient. The choices are identical to those by (10, 11) for temporally not extended tasks. Specifically the results in ref. (11) appear as special case of our results for multiple input patterns, if the task-dimension is maximal, single trials have no temporal extent and the inputs have fluctuating amplitude (SN 5). We choose the NP scheme such that it matches the WP scheme. It is a discrete time version of the NP scheme proposed by (12) and an extension of the scheme by (11) to temporally extended tasks. In biologically plausible implementations of WP and NP, the reward should be compared to an intrinsically generated prediction, such as an average of previous rewards (14–16) or the reward of another perturbed trial (64). In the delayed non-match-to-sample task, we thus replace our standard unperturbed baseline by such an average. This also allows a direct comparison with the NP scheme by (16).

NP is studied in various concrete neurobiological settings. Previous work used feedforward networks with NP to model the learning of coordinate transforms in the visual system (65), birdsong (18, 50) and motor output (14, 66). (15) show that reservoir computers with NP trained, fed back readouts can learn periodic inputs, routing and working memory tasks. (16) uses a fully plastic recurrent network for the learning of a delayed non-match-to-sample, a selective integration and a motor control task. Finally, NP is often employed for reference and comparison (67–71). WP is considered less in studies of neurobiological learning. It is implemented in early feedforward network models for birdsong (72) and binary output task learning (55, 63). Further, it is occasionally used for comparison (68, 70). Very recently (4) have shown that recurrent neural networks can be pretrained with WP and the reservoir computing scheme to thereafter learn with static weights to generate fixed point activity.

The results of our present article using feedforward, reservoir computing and fully plastic recurrent networks suggest that for many neurobiological tasks WP is at least as suitable as NP, while the neurobiological implementation may be even simpler. This indicates that WP is a similarly plausible model for learning in the brain as NP. Experimentally measurable learning and weight dynamics allow to distinguish the learning rules.

Methods

Analytical error dynamics. To analytically compute the dynamics of the expected error, we consider an arbitrary perturbation ξ. This determines the error change $E^{\text{pert}} - E$ and the resulting weight update Δw via Eqs. (3,6)). Δw in turn determines the new weights and via Eq. (8) the error $E(n+1)$ after the update. $E(n+1)$ is thus a function of ξ, the weight mismatch $W(n)$ before the update and the input correlations S,

$$E(n+1) = \tfrac{1}{2}\,\mathrm{tr}[(W(n)+\Delta w(\xi))\; S\; (W(n)+\Delta w(\xi))^T]. \tag{17}$$

Averaging over perturbations and using Isserlis' theorem yields an equation for the expected error $\langle E(n+1)\rangle$. When assuming that all latent inputs have the same strength, $\langle E(n+1)\rangle$ becomes a function of the error $\langle E(n)\rangle$ before the update and the system parameters, leading to Eq. (9). The detailed derivation is given in SN 2.

Numerical simulations accompanying the theoretical analysis. In the numerical experiments in sec. "Theoretical Analysis", the N_{eff} nonzero inputs are orthonormal functions, superpositions of sines, scaled by $\alpha^2 = N/N_{\text{eff}}$ to keep the total input strength $\alpha^2 N_{\text{eff}}$ for different N_{eff} constant. Targets z_{it} are obtained by linearly combining these functions using teacher weights $w^*_{ij} = 0.1$ and adding as unrealizable component a further, appropriately scaled, orthonormal function. Learning rates are η^*.

Reservoir computing task. The $N = 500$ rate neurons of the fully connected recurrent reservoir network evolve according to

$$x_{jt} = \gamma x_{j,t-1} + (1-\gamma)\Big(\sum_{k=1}^{N} w^{\text{rec}}_{jk} r_{k,t-1} + \sum_{q=1}^{N_{\text{inputs}}} w^{\text{in}}_{jq} r^{\text{in}}_{qt}\Big). \tag{18}$$

The rate of neuron k is $r_{kt} = \tanh(x_{kt})$. Their decay time constant is $\tau = 10$ time steps, i.e. $\gamma = \mathrm{e}^{-1/\tau}$. Recurrent weights w^{rec} are drawn from a centered normal distribution, the weight matrix is thereafter normalized to ensure that the real part of its largest eigenvalue is $g_{\text{rec}} = 1$. Input weights w^{in} are drawn from $w^{\text{in}}_{jk} \sim \mathcal{N}(0, 1/N_{\text{in}})$. Creating various instances of such random networks showed that performance and participation ratio $\mathrm{PR} = (\sum_{\mu=1}^{N} \alpha_\mu^2)^2 / \sum_{\mu=1}^{N} \alpha_\mu^4$ are rather independent of the instance. The participation ratio gives an estimate of the dimensionality of the reservoir dynamics (20, 73). Generally we observe $\mathrm{PR} \approx 5$, for example in the network of Fig. 5 $\mathrm{PR} \approx 5.3$. The $N_{\text{in}} = 5$ inputs to the reservoir are orthogonal to each other, $r^{\text{in}}_{1t} = 1$, $r^{\text{in}}_{2t} = \sqrt{2}\sin(\omega t)$, $r^{\text{in}}_{3t} = \sqrt{2}\cos(\omega t)$, $r^{\text{in}}_{4t} = \sqrt{2}\sin(2\omega t)$, $r^{\text{in}}_{5t} = \sqrt{2}\cos(2\omega t)$, $\omega = 2\pi/T$, $T = 500$ timesteps. The trained linear readout produces $M = 2$ outputs $z_{it} = \sum_{j=1}^{N} w_{ij} r_{jt}$. Their target z^*_t is, up to scaling, the same as in (40): $z^*_{1t} = \mathrm{radius}_t \cos(\omega t)$, $z^*_{2t} = \mathrm{radius}_t \sin(\omega t)$, with $\mathrm{radius}_t = 0.1 \cdot (9 - \sin(\omega t) + 2\sin(3\omega t) + 2\sin(5\omega t) - \sin(7\omega t) + 3\cos(2\omega t) - 2\cos(4\omega t))$. Already 100 timesteps before the task starts, the reservoir is initialized and given external input. By the time the task begins, network activity is enslaved by the external input and has settled down to a periodic orbit. Technically, we record the reservoir activity traces r_{jt} once for the entire training of w, because they are the same in each trial. The value of the participation ratio motivated us to construct an optimal readout reading out the largest five principal components via the least squares fit $w^{\text{LS}}_{ij} = \sum_{k=1}^{N} \sum_{t=1}^{T} z^*_{it} r_{kt} S^{\text{pinv}}_{kj}$ (Fig. 5b, dashed gray). Here

S^{pinv} is the pseudoinverse of the reduced correlation matrix of the reservoir that is obtained by setting all eigenvalues of S except the largest five to zero. Including six principal components did not qualitatively change the result.
From the theoretical analysis Eq. (12), we obtain an estimate $\eta^* = 1/((M\mathrm{PR}+2)\overline{\alpha^2})$ for the optimal learning rate, by setting $N_{\mathrm{eff}} \to \mathrm{PR}$ and $\alpha^2 \to \overline{\alpha^2} = \frac{1}{\mathrm{PR}}\sum_{\mu=1}^{N}\alpha_\mu^2$. $\overline{\alpha^2}$ is the strength of each latent input when we assume that the total input strength is generated by PR equally strong ones. We verified by a grid search that this estimated value yields close to optimal performance for both WP and NP and chose it as the learning rate. A scan over σ_{eff} confirmed that the final error depends quadratically on it, as predicted by the theory.

Delayed non-match-to-sample task. The fully connected recurrent network has $N = 200$ rate neurons. The dynamics of neuron i, $i = 4,\ldots,N$, are governed by

$$\tau\dot{x}_i = -x_i(t) + \sum_{j=1}^{N} w_{ij}^{\mathrm{rec}} r_j(t) + \sum_{q=1}^{2} w_{iq}^{\mathrm{in}} u_q(t), \qquad \textbf{(19)}$$

with time constant $\tau = 30\,\mathrm{ms}$. The constant activations $x_1(t) = x_2(t) = 1$ and $x_3(t) = -1$ provide biases (16). The rate of each neuron i, $i = 1,\ldots,N$, is given by $r_i(t) = \tanh(x_i(t))$. $z(t) = r_4(t)$ is the network output. We use the forward Euler-method with stepsize $dt = 1\,\mathrm{ms}$ to simulate the dynamics and draw the initial activations from a uniform distribution, $x_i(0) \sim \mathcal{U}(-0.1, 0.1)$ for $i = 4,\ldots,N$. Recurrent weights are drawn from a Gaussian distribution, $w_{ij}^{\mathrm{rec}} \sim \mathcal{N}(0, g^2/N)$, with $g = 1.5$. Input weights are drawn from a uniform distribution, $w_{iq}^{\mathrm{in}} \sim \mathcal{U}(-1, 1)$.
All recurrent weights w_{ij}^{rec} are trained. The error function of WP and NP is the mean squared difference between the output z and the target within the last $200\,\mathrm{ms}$ of each trial. For each of the different trial types k, $k = 1,\ldots,4$, we use an exponential average of the previous errors $E^{\mathrm{pert}}(n_{\mathrm{k}})$ for this trial type (n_{k} indexes the trials of type k) as the error baseline:

$$E_{\mathrm{k}}(n_{\mathrm{k}}) = E_{\mathrm{k}}(n_{\mathrm{k}}-1) + \frac{1}{\tau_{\mathrm{E}}}\left(E^{\mathrm{pert}}(n_{\mathrm{k}}) - E_{\mathrm{k}}(n_{\mathrm{k}}-1)\right), \qquad \textbf{(20)}$$

where $\tau_{\mathrm{E}} = 4$. To get the best performing learning parameters, we performed a grid search, which yielded $\eta^{\mathrm{WP}} = 1\times10^{-5}, \sigma^{\mathrm{WP}} = 4.64\times10^{-3}, \eta^{\mathrm{NP}} = 1\times10^{-5}, \sigma^{\mathrm{NP}} = 4.64\times10^{-1}$.
For the details of the version of NP proposed by (16), see this article. For the convenience of the reader here we briefly mention the main differences to the vanilla NP version Eq. (6): For each network neuron a node perturbation is applied at a simulation time step only with a probability of $0.3\,\%$ and is drawn from a uniform distribution, $\xi \sim \mathcal{U}(-16, 16)$. The error is given by the absolute difference between output and target. Weight updates are computed via $\Delta w_{ij}^{\mathrm{rec}}(n_k) = -\eta E_k(n_k - 1)\left(E^{\mathrm{pert}}(n_k) - E_k(n_k-1)\right)\sum_{t=1}^{T}\left[(x_{it} - \bar{x}_{it})\, r_{j,t-1}\right]^3$, and clipped when they exceed $\pm 3\times10^{-4}$ (cf. code accompanying (16)). t indexes the simulation time step of each trial, T is the total number of simulation time steps per trial and $\bar{x}_{it} = \bar{x}_{i,t-1} + \frac{1}{\tau_{\mathrm{x}}}(x_{it} - \bar{x}_{i,t-1})$ is an exponential average of past activations. Parameter values are $\eta = 0.1, \tau_x = \frac{20}{19}$.

MNIST classification task. The input layer of the fully connected feedforward network consists of 784 units encoding the pixel values of the data. The hidden layer consists of 100 neurons with $\tanh$ activation function and biases. The output layer consists of 10 neurons, one for each single-digit number, with softmax activation function and biases. We use the standard training and test data set, but split the standard training data set into a training data set of $50\,000$ images and a validation data set of $10\,000$ images. No preprocessing is done on the data. We employ vanilla WP Eq. (3), NP Eq. (6) or SGD to train all parameters of the network. The error function is the cross-entropy loss averaged over the batch of length $N_{\mathrm{batch}} = T$. We also tried to combine the gradient estimates obtained from WP and NP with Momentum, RMSProp or Adam (28), but did not find an improvement of performance compared to the vanilla versions with carefully tuned parameters. The same holds for SGD. This may be because of the rather simple network architecture.
To obtain the best-performing parameters (the learning rate for all three algorithms and the standard deviation for WP and NP), we performed a grid search for each of the considered batch sizes: For each parameter set we trained the network for $50\,000$ trials (i.e.: weight updates) on the training data set. We then selected the best-performing parameter sets based on the final accuracy on the validation data set and applied them to the test data set. High final accuracy appeared to concur with fast convergence speed, such that a comparison to our analytical results (where learning rate optimizes the convergence speed) seems justified.

ACKNOWLEDGEMENTS
We thank the German Federal Ministry of Education and Research (BMBF) for support via the Bernstein Network (Bernstein Award 2014, 01GQ1710).

Bibliography

1. Daniel M. Wolpert, Jörn Diedrichsen, and J. Randall Flanagan. Principles of sensorimotor learning. *Nat. Rev. Neurosci.*, 12(12):739–751, October 2011. doi: 10.1038/nrn3112.
2. John W Krakauer and Pietro Mazzoni. Human sensorimotor learning: Adaptation, skill, and beyond. *Curr. Opin. Neurobiol.*, 21:636–644, August 2011. ISSN 0959-4388. doi: 10.1016/j.conb.2011.06.012.
3. Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nat. Rev. Neurosci.*, 21(6):335–346, April 2020. doi: 10.1038/s41583-020-0277-3.
4. Christian Klos, Yaroslav Felipe Kalle Kossio, Sven Goedeke, Aditya Gilra, and Raoul-Martin Memmesheimer. Dynamical learning of dynamics. *Physical Review Letters*, 125(8), August 2020. doi: 10.1103/physrevlett.125.088103.
5. Ashesh K. Dhawale, Maurice A. Smith, and Bence P. Ölveczky. The role of variability in motor learning. *Annual Review of Neuroscience*, 40(1):479–498, July 2017. doi: 10.1146/annurev-neuro-072116-031548.
6. R. Mooney, J. Prather, and T. Roberts. Neurophysiology of birdsong learning. In *Learning and Memory: A Comprehensive Reference*, pages 441–474. Elsevier, 2008. doi: 10.1016/b978-012370509-9.00116-9.
7. P. Dayan and L. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* MIT Press, Cambridge, 2001.
8. Richard S. Sutton. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning series).* A Bradford Book, nov 2018. ISBN 0262039249.
9. A. Dembo and T. Kailath. Model-free distributed learning. *IEEE Transactions on Neural Networks*, 1(1):58–70, mar 1990. ISSN 10459227. doi: 10.1109/72.80205.
10. G. Cauwenberghs. A Fast Stochastic Error-Descent Algorithm for Supervised Learning and Optimization. In *Advances in Neural Information Processing Systems*, volume 5, pages 244–251, mar 1993.
11. Justin Werfel, Xiaohui Xie, and H. Sebastian Seung. Learning curves for stochastic gradient descent in linear feedforward networks. *Neural Comput.*, 17(12):2699–2718, December 2005. doi: 10.1162/089976605774320539.

12. I. R. Fiete and H. S. Seung. Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Phys. Rev. Lett.*, 97(4):1–4, 2006. ISSN 00319007. doi: 10.1103/PhysRevLett.97.048104.
13. Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.
14. R. Legenstein, S. M. Chase, A. B. Schwartz, and W. Maass. A reward-modulated hebbian learning rule can explain experimentally observed network reorganization in a brain control task. *J. Neurosci.*, 30(25):8400–8410, June 2010. doi: 10.1523/jneurosci.4284-09.2010.
15. Gregor M. Hoerzer, Robert Legenstein, and Wolfgang Maass. Emergence of complex computational structures from chaotic neural networks through reward-modulated hebbian learning. *Cereb. Cortex*, 24:677–690, 2014. doi: 10.1093/cercor/bhs348.
16. T. Miconi. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife*, 6:e20899, 2017. ISSN 2050-084X. doi: 10.7554/elife.20899.
17. M. Jabri and B. Flower. Weight perturbation: an optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayer networks. *IEEE Transactions on Neural Networks*, 3(1):154–157, 1992. ISSN 10459227. doi: 10.1109/72.105429.
18. Ila R Fiete, Michale S Fee, and H. Sebastian Seung. Model of birdsong learning based on gradient estimation by dynamic perturbation of neural conductances. *J. Neurophysiol.*, 98: 2038–2057, 2007. doi: 10.1152/jn.01311.2006.
19. Hiroshi Saito, Kentaro Katahira, Kazuo Okanoya, and Masato Okada. Statistical mechanics of structural and temporal credit assignment effects on learning in neural networks. *Phys. Rev. E*, 83(5):051125, May 2011. ISSN 1539-3755, 1550-2376. doi: 10.1103/PhysRevE.83.051125.
20. Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. bioRxiv, November 2017.
21. Matthew C Tresch and Anthony Jarc. The case for and against muscle synergies. *Current Opinion in Neurobiology*, 19(6):601–607, December 2009. doi: 10.1016/j.conb.2009.09.002.
22. Juan A. Gallego, Matthew G. Perich, Lee E. Miller, and Sara A. Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, June 2017. doi: 10.1016/j.neuron.2017.05.025.
23. Kenji Doya and Terrence J. Sejnowski. A computational model of birdsong learning by auditory experience and auditory feedback. In *Central Auditory Processing and Neural Modeling*, pages 77–88. Springer US, 1998. doi: 10.1007/978-1-4615-5351-9_8.
24. Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics - From single neurons to networks and models of cognition*. Cambridge University Press, Cambridge, 2014.
25. B. Widrow and M. A. Lehr. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990. ISSN 00189219. doi: 10.1109/5.58323.
26. M. London and M. Häusser. Dendritic computation. *Annu. Rev. Neurosci.*, 28:503–532, 2005.
27. Raoul-Martin Memmesheimer. Quantitative prediction of intermittent high-frequency oscillations in neural networks with supralinear dendritic interactions. *Proc. Natl Acad. Sci. USA*, 107:11092–11097, 2010. doi: 10.1073/pnas.0909615107.
28. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
29. Friedemann Zenke, Guillaume Hennequin, and Wulfram Gerstner. Synaptic plasticity in neural networks needs homeostasis with a fast rate detector. *PLOS Comp. Biol.*, 9(11):e1003330, November 2013. doi: 10.1371/journal.pcbi.1003330.
30. Noam E. Ziv and Naama Brenner. Synaptic tenacity or lack thereof: Spontaneous remodeling of synapses. *Trends Neurosci.*, 41(2):89–99, February 2018. doi: 10.1016/j.tins.2017.12.003.
31. Risa Kawai, Timothy Markman, Rajesh Poddar, Raymond Ko, Antoniu L. Fantana, Ashesh K. Dhawale, Adam R. Kampff, and Bence P. Ölveczky. Motor cortex is required for learning but not for executing a motor skill. *Neuron*, 86(3):800–812, May 2015. doi: 10.1016/j.neuron.2015.03.024.
32. Michael Beyeler, Emily L. Rounds, Kristofor D. Carlson, Nikil Dutt, and Jeffrey L. Krichmar. Neural correlates of sparse coding and dimensionality reduction. *PLOS Computational Biology*, 15(6):e1006908, June 2019. doi: 10.1371/journal.pcbi.1006908.
33. R. Hahnloser, A. Kozhevnikov, and M. Fee. An ultra-sparse code underlies the generation of neural sequences in a songbird. *Nature*, 419:65–70, 2002.
34. R Quian Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435:1102–1107, Jun 2005. doi: 10.1038/nature03687.
35. D. V. Buonomano and M. M. Merzenich. Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, 267:1028–1030, 1995.
36. P. F. Dominey. Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. *Biol. Cybern.*, 73:265–274, Aug 1995.
37. W. Maass, T. Natschläger, and H. Markram. A model for real time computation in generic microcircuits. In S. Becker, S. Thrün, and K. Obermayer, editors, *Advances in neural information processing systems*. MIT Press, 2003.
38. David Sussillo and L. F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63:544–557, 2009. doi: 10.1016/j.neuron.2009.07.018.
39. Dominik Thalmeier, Marvin Uhlmann, Hilbert J. Kappen, and Raoul-Martin Memmesheimer. Learning universal computations with spikes. *PLOS Comput. Biol.*, 12:e1004895, 2016.
40. Ryan Pyle and Robert Rosenbaum. A reservoir computing model of reward-modulated motor learning and automaticity. *Neural Computation*, 31(7):1430–1461, July 2019. doi: 10.1162/neco_a_01198.
41. Guillaume Hennequin, Tim P. Vogels, and Wulfram Gerstner. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron*, 82:1394–1406, Jun 2014. doi: 10.1016/j.neuron.2014.04.045.
42. M. F. Bear, B. W. Connors, and M. A. Paradiso. *Neuroscience - Exploring the Brain*. Wolters Kluwer, Philadelphia, 2016. ISBN 9780781778176.
43. Nicolas Y Masse, Guangyu R Yang, H Francis Song, Xiao-Jing Wang, and David J Freedman. Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nature Neuroscience*, 22:1159—-1167, 2019. doi: 10.1038/s41593-019-0414-3.
44. John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, August 2014. doi: 10.1038/nn.3776.
45. Larry F Abbott, Kanaka Rajan, and Haim Sompolinsky. *The Dynamic brain: an exploration of neuronal variability and its functional significance*, chapter Interactions between Intrinsic and Stimulus-Evoked Activity in Recurrent Neural Networks, pages 65–82. Oxford Univ. Press, Oxford, 2011.
46. Christopher J. Cueva, Alex Saez, Encarni Marcos, Aldo Genovesio, Mehrdad Jazayeri, Ranulfo Romo, C. Daniel Salzman, Michael N. Shadlen, and Stefano Fusi. Low-dimensional dynamics for working memory and time encoding. *Proceedings of the National Academy of Sciences*, 117(37):23021–23032, August 2020. doi: 10.1073/pnas.1915984117.
47. Marta Russo, Mattia D'Andola, Alessandro Portone, Francesco Lacquaniti, and Andrea d'Avella. Dimensionality of joint torques and muscle patterns for reaching. *Frontiers in Computational Neuroscience*, 8, 2014. doi: 10.3389/fncom.2014.00024.
48. Alain Destexhe, Michael Rudolph, and Denis Paré. The high-conductance state of neocortical neurons in vivo. *Nat Rev Neurosci*, 4:739–751, Sep 2003. doi: 10.1038/nrn1198.
49. John D Murray, Alberto Bernacchia, David J Freedman, Ranulfo Romo, Jonathan D Wallis, Xinying Cai, Camillo Padoa-Schioppa, Tatiana Pasternak, Hyojung Seo, Daeyeol Lee, and Xiao-Jing Wang. A hierarchy of intrinsic timescales across primate cortex. *Nature Neuroscience*, 17(12):1661–1663, November 2014. doi: 10.1038/nn.3862.
50. Tiberiu Teşileanu, Bence Ölveczky, and Vijay Balasubramanian. Rules and mechanisms for efficient two-stage learning in neural circuits. *eLife*, 6, April 2017. doi: 10.7554/elife.20944.
51. James M Murray and G Sean Escola. Learning multiple variable-speed sequences in striatum via cortical tutoring. *eLife*, 6, May 2017. doi: 10.7554/elife.26084.
52. Ashesh K Dhawale, Rajesh Poddar, Steffen BE Wolff, Valentin A Normand, Evi Kopelowitz, and Bence P Ölveczky. Automated long-term recording and analysis of neural activity in behaving animals. *eLife*, 6, September 2017. doi: 10.7554/elife.27702.
53. Sen Cheng. The CRISP theory of hippocampal function in episodic memory. *Front. Neural Circuits*, 7:88, 2013. doi: 10.3389/fncir.2013.00088.
54. Amadeus Maes, Mauricio Barahona, and Claudia Clopath. Learning spatiotemporal signals using a recurrent spiking network that discretizes time. *PLOS Computational Biology*, 16(1): e1007606, January 2020. doi: 10.1371/journal.pcbi.1007606.
55. H. S. Seung. Learning in Spiking Neural Networks by Reinforcement of Stochastic Synaptic Transmission. *Neuron*, 40(6):1063–1073, dec 2003. ISSN 08966273. doi: 10.1016/S0896-6273(03)00761-X.
56. Łukasz Kuśmierz, Takuya Isomura, and Taro Toyoizumi. Learning with three factors: modulating hebbian plasticity with errors. *Current Opinion in Neurobiology*, 46:170–177, October 2017. doi: 10.1016/j.conb.2017.08.020.
57. R. L. Redondo and R. G. M. Morris. Making memories last: the synaptic tagging and capture hypothesis. *Nat. Rev. Neurosci.*, 12(1):17–30, Dec 2011. doi: 10.1038/nrn2963.
58. John N.J Reynolds and Jeffery R Wickens. Dopamine-dependent plasticity of corticostriatal synapses. *Neural Networks*, 15(4-6):507–521, June 2002. doi: 10.1016/s0893-6080(02)00045-x.
59. Qiang Zhou and Muming Poo. Reversal and consolidation of activity-induced synaptic modifications. *Trends in Neurosciences*, 27(7):378–383, July 2004. doi: 10.1016/j.tins.2004.05.006.
60. Jannik Luboeinski and Christian Tetzlaff. Memory consolidation and improvement by synaptic tagging and capture in recurrent neural networks. *Communications Biology*, 4(1), March 2021. doi: 10.1038/s42003-021-01778-y.
61. E. M. Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 17(10):2443–2452, January 2007. doi: 10.1093/cercor/bhl152.
62. Michael A. Farries and Adrienne L. Fairhall. Reinforcement learning with modulated spike timing–dependent synaptic plasticity. *Journal of Neurophysiology*, 98(6):3648–3665, December 2007. doi: 10.1152/jn.00364.2007.
63. Piotr Suszynski and Pawel Wawrzynski. Learning population of spiking neural networks with perturbation of conductances. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, August 2013. doi: 10.1109/ijcnn.2013.6706756.
64. Tatsuya Cho, Kentaro Katahira, Kazuo Okanoya, and Masato Okada. Node perturbation learning without noiseless baseline. *Neural Networks*, 24(3):267–272, April 2011. doi: 10.1016/j.neunet.2010.12.001.
65. P. Mazzoni, R. A. Andersen, and M. I. Jordan. A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10):4433–4437, May 1991. doi: 10.1073/pnas.88.10.4433.
66. Ran Darshan, Arthur Leblois, and David Hansel. Interference and shaping in sensorimotor adaptations with rewards. *PLoS Computational Biology*, 10(1):e1003377, January 2014. doi: 10.1371/journal.pcbi.1003377.
67. Eleni Vasilaki, Stefano Fusi, Xiao-Jing Wang, and Walter Senn. Learning flexible sensori-motor mappings in a complex network. *Biological Cybernetics*, 100(2):147–158, January 2009. doi: 10.1007/s00422-008-0288-z.
68. Ken Takiyama and Masato Okada. Maximization of learning speed in the motor cortex due to neuronal redundancy. *PLoS Computational Biology*, 8(1):e1002348, January 2012. doi: 10.1371/journal.pcbi.1002348.
69. M. N. Abdelghani, T. P. Lillicrap, and D. B. Tweed. Sensitivity derivatives for flexible sensorimotor learning. *Neural Computation*, 20(8):2085–2111, August 2008. doi: 10.1162/neco.2008.04-07-507.
70. Boris B. Vladimirskiy, Eleni Vasilaki, Robert Urbanczik, and Walter Senn. Stimulus sampling as an exploration mechanism for fast reinforcement learning. *Biological Cybernetics*, 100(4): 319–330, April 2009. doi: 10.1007/s00422-009-0305-x.
71. Johannes Friedrich, Robert Urbanczik, and Walter Senn. Code-specific learning rules improve action selection by populations of spiking neurons. *Int J Neural Syst*, 24:1450002, Aug 2014. doi: 10.1142/S0129065714500026.
72. K. Doya. Bifurcations in the learning of recurrent neural networks. *Proc. IEEE Intl Symp. Circuits Systems* 6:2777-2780, 1992.
73. Marco Del Giudice. Effective dimensionality: A tutorial. *Multivariate Behavioral Research*, 56 (3):527–542, 2021. doi: 10.1080/00273171.2020.1743631. PMID: 32223436.