Credit Assignment in Neural Networks through Deep Feedback Control

Anonymous Author(s)
Affiliation
Address
email

Abstract

The success of deep learning sparked interest in whether the brain learns by using similar techniques for assigning credit to each synaptic weight for its contribution to the network output. However, the majority of current attempts at biologically-plausible learning methods are either non-local in time, require highly specific connectivity motives, or have no clear link to any known mathematical optimization method. Here, we introduce Deep Feedback Control (DFC), a new learning method that uses a feedback controller to drive a deep neural network to match a desired output target and whose control signal can be used for credit assignment. The resulting learning rule is fully local in space and time and approximates Gauss-Newton optimization for a wide range of feedback connectivity patterns. To further underline its biological plausibility, we relate DFC to a multi-compartment model of cortical pyramidal neurons with a local voltage-dependent synaptic plasticity rule, consistent with recent theories of dendritic processing. By combining dynamical system theory with mathematical optimization theory, we provide a strong theoretical foundation for DFC that we corroborate with detailed results on toy experiments and standard computer-vision benchmarks.

1 Introduction

The error backpropagation (BP) algorithm [1, 2, 3] is currently the gold standard to perform credit assignment (CA) in deep neural networks. Although deep learning was inspired by biological neural networks, an exact mapping of BP onto biology to explain learning in the brain leads to several inconsistencies with experimental results that are not yet fully addressed [4, 5, 6]. First, BP requires an exact symmetry between the weights of the forward and feedback pathways [5, 6], also called the weight transport problem. Another issue of relevance is that, in biological networks, feedback also changes each neuron's activation and thus its immediate output [7, 8], which does not occur in BP.

Lillicrap et al. [9] convincingly showed that the weight transport problem can be sidestepped in modest supervised learning problems by using random feedback connections. However, follow-up studies indicated that random feedback paths cannot provide precise CA in more complex problems [10, 11, 12, 13], which can be mitigated by learning feedback weights that align with the forward pathway [14, 15, 16, 17] or approximate its inverse [18, 19, 20, 21]. However, this precise alignment imposes strict constraints on the feedback weights, whereas more flexible constraints could provide the freedom to use feedback also for other purposes besides learning, such as attention and prediction [8].

A complementary line of research proposes models of cortical microcircuits which propagate CA signals through the network using dynamic feedback [22, 23, 24] or multiplexed neural codes [25], thereby directly influencing neural activations with feedback. However, these models introduce highly specific connectivity motifs and tightly coordinated plasticity mechanisms. Whether these

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

constraints can be fulfilled by cortical networks is an interesting experimental question. Another line of work uses adaptive control theory [26] to derive learning rules for non-hierarchical recurrent neural networks (RNNs) based on error feedback, which drives neural activity to track a reference output [27, 28, 29, 30]. These methods have so far only been used to train single-layer RNNs with fixed output and feedback weights, making it unclear whether they can be extended to deep neural networks. Finally, two recent studies [31, 32] use error feedback in a dynamical setting to invert the forward pathway, thereby enabling errors to flow backward. These approaches rely on a learning rule that is non-local in time and it remains unclear whether they approximate any known optimization method. Addressing the latter, two recent studies take a first step by relating learned (non-dynamical) inverses of the forward pathway [20] and iterative inverses restricted to invertible networks [21] to approximate Gauss-Newton optimization.

Inspired by the Dynamic Inversion method [31], we introduce Deep Feedback Control (DFC), a new biologically-plausible CA method that addresses the above-mentioned limitations and extends the control theory approach to learning [27, 28, 29, 30] to deep neural networks. DFC uses a feedback controller that drives a deep neural network to match a desired output target. For learning, DFC then simply uses the dynamic change in the neuron activations to update their synaptic weights, resulting in a learning rule fully local in space and time. We show that DFC approximates Gauss-Newton (GN) optimization and therefore provides a fundamentally different approach to CA compared to BP. Furthermore, DFC does not require precise alignment between forward and feedback weights, nor does it rely on highly specific connectivity motives. Interestingly, the neuron model used by DFC can be closely connected to recent multi-compartment models of cortical pyramidal neurons. Finally, we provide detailed experimental results, corroborating our theoretical contributions and showing that DFC does principled CA on standard computer-vision benchmarks in a way that fundamentally differs from standard BP.

2 The Deep Feedback Control method

Here, we introduce the core parts of DFC. In contrast to conventional feedforward neural network models, DFC makes use of a dynamical neuron model (Section 2.1). We use a feedback controller to drive the neurons of the network to match a desired output target (Section 2.2), while simultaneously updating the synaptic weights using the change in neuronal activities (Section 2.3). This combination of dynamical neurons and controller leads to a simple but powerful learning method, that is linked to GN optimization and offers a flexible range of feedback connectivity (see Section 3).

2.1 Neuron and network dynamics

The first main component of DFC is a dynamical multilayer network, in which every neuron integrates its forward and feedback inputs according to the following dynamics:

$$\tau_v \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}_i(t) = -\mathbf{v}_i(t) + W_i\phi\big(\mathbf{v}_{i-1}(t)\big) + Q_i\mathbf{u}(t) \quad 1 \leq i \leq L, \tag{1}$$

with $\mathbf{v}_i$ a vector containing the pre-nonlinearity activations of the neurons in layer i, W_i the forward weight matrix, ϕ a smooth nonlinearity, $\mathbf{u}$ a feedback input, Q_i the feedback weight matrix, and τ_v a time constant. See Fig. 1B for a schematic representation of the network. To simplify notation, we define $\mathbf{r}_i = \phi(\mathbf{v}_i)$ as the post-nonlinearity activations of layer i. The input $\mathbf{r}_0$ remains fixed throughout the dynamics (1). Note that in the absence of feedback, i.e., $\mathbf{u} = 0$, the equilibrium state of the network dynamics (1) corresponds to a conventional multilayer feedforward network state, which we denote with superscript '-':

$$\mathbf{r}_i^- = \phi(\mathbf{v}_i^-) = \phi(W_i\mathbf{r}_{i-1}^-), \quad 1 \leq i \leq L, \quad \text{with } \mathbf{r}_0^- = \mathbf{r}_0. \tag{2}$$

2.2 Feedback controller

The second core component of DFC is a feedback controller, which is only active during learning. Instead of a single backward pass for providing feedback, DFC uses a feedback controller to continuously drive the network to an output target $\mathbf{r}_L^*$ (see Fig. 1D). Following the Target Propagation framework [19, 20, 21], we define $\mathbf{r}_L^*$ as the feedforward output nudged towards lower loss:

$$\mathbf{r}_L^* \triangleq \mathbf{r}_L^- - \lambda \frac{\partial \mathcal{L}(\mathbf{r}_L, \mathbf{y})}{\partial \mathbf{r}_L}\Big|_{\mathbf{r}_L = \mathbf{r}_L^-} = \mathbf{r}_L^- + \boldsymbol{\delta}_L, \tag{3}$$

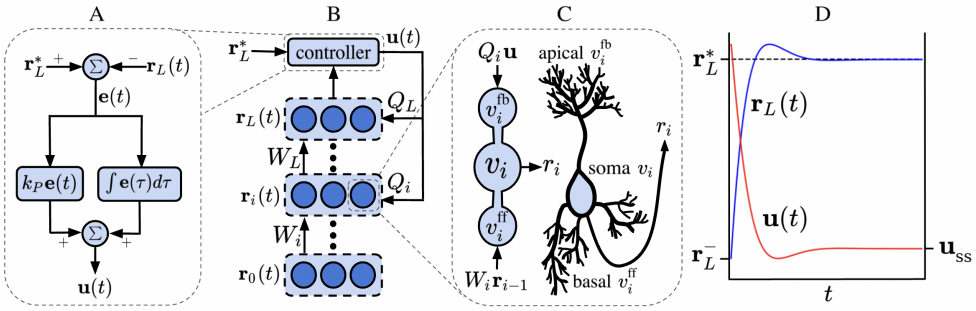

Figure 1: (A) A block diagram of the controller, where we omitted the leakage term of the integral controller. (B) Schematic illustration of DFC. (C) Schematic illustration of the multi-compartment neuron used by DFC, compared to a cortical pyramidal neuron sketch (see also Discussion). (D) Illustration of the output $\mathbf{r}_L(t)$ and the controller dynamics $\mathbf{u}(t)$ in DFC.

with $\mathcal{L}(\mathbf{r}_L, \mathbf{y})$ a supervised loss function defining the task, $\mathbf{y}$ the label of the training sample, λ a stepsize, and $\boldsymbol{\delta}_L$ shorthand notation. Note that (3) only needs the easily obtained loss gradient w.r.t. the output, e.g., for an L^2 output loss, one obtains the convex combination $\mathbf{r}_L^* = (1 - 2\lambda)\mathbf{r}_L^- + 2\lambda\mathbf{y}$.

The feedback controller produces a feedback signal $\mathbf{u}(t)$ to drive the network output $\mathbf{r}_L(t)$ towards its target $\mathbf{r}_L^*$, using the control error $\mathbf{e}(t) \triangleq \mathbf{r}_L^* - \mathbf{r}_L(t)$. A standard approach in designing a feedback controller is the Proportional-Integral-Derivative (PID) framework [33]. While DFC is compatible with a full PID controller, we only use the integral and proportional part for simplicity, resulting in the following feedback controller (see also Fig. 1A):

$$\mathbf{u}(t) = K_I\mathbf{u}^{\text{int}}(t) + K_P\mathbf{e}(t), \quad \tau_u \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{u}^{\text{int}}(t) = \mathbf{e}(t) - \alpha\mathbf{u}^{\text{int}}(t), \tag{4}$$

where a leakage term is added to constrain the magnitude of $\mathbf{u}^{\text{int}}$. For mathematical simplicity, we take the control matrices equal to $K_I = I$ and $K_P = k_p I$ with $k_p \geq 0$ the proportional control constant. This controller adds a leaky integration of the error $\mathbf{u}^{\text{int}}$ to a scaled version of the error $k_p\mathbf{e}$ which could be implemented by a dedicated neural microcircuit (for a discussion see App. H). Drawing inspiration from the Target Propagation framework [18, 19, 20, 21] and the Dynamic Inversion framework [31], one can think of the controller as performing a *dynamic inversion* of the output target $\mathbf{r}_L^*$ towards the hidden layers, as it dynamically changes the activation of the hidden layers until the output target is reached.

2.3 Forward weight updates

The update rule for the feedforward weights has the form:

$$\tau_W \frac{\mathrm{d}}{\mathrm{d}t} W_i(t) = \big(\phi(\mathbf{v}_i(t)) - \phi(W_i\mathbf{r}_{i-1}(t))\big)\mathbf{r}_{i-1}(t)^T. \tag{5}$$

This learning rule simply compares the neuron's controlled activation to its current feedforward input and is thus local in space and time. Furthermore, it can be interpreted most naturally by compartmentalizing the neuron into the central compartment $\mathbf{v}_i$ from (1) and a feedforward compartment $\mathbf{v}_i^{\text{ff}} \triangleq W_i\mathbf{r}_{i-1}$ that integrates the feedforward input. Now, the forward weight dynamics (5) represents a delta rule using the difference between the actual firing rate of the neuron, $\phi(\mathbf{v}_i)$, and its estimated firing rate, $\phi(\mathbf{v}_i^{\text{ff}})$, based on the feedforward inputs. Note that we assume τ_W to be a large time constant, such that the network (1) and controller dynamics (4) are not influenced by the weight dynamics, i.e., the weights are considered fixed in the timescale of the controller and network dynamics.

In Section 5, we show how the feedback weights Q_i can also be learned locally in time and space for supporting the stability of the network dynamics and the learning of W_i. This feedback learning rule needs a feedback compartment $\mathbf{v}_i^{\text{fb}} \triangleq Q_i\mathbf{u}$, leading to the *three-compartment neuron* schematized in Fig. 1C, inspired by recent multi-compartment models of the pyramidal neuron (see Discussion). Now that we introduced the DFC model, we will show that (i) the weight updates (5) can properly optimize a loss function (Section 3), (ii) the resulting dynamical system is stable under certain conditions (Section 4), and (iii) learning the feedback weights facilitates (i) and (ii) (Section 5).

3 Learning theory

To understand how DFC optimizes the feedforward mapping (2) on a given loss function, we link the weight updates (5) to mathematical optimization theory. We start by showing that DFC dynamically inverts the output error to the hidden layers (Section 3.1), which we link to GN optimization under flexible constraints on the feedback weights Q_i and on layer activations (Section 3.2). In Section 3.3, we relax some of these constraints, and show that DFC still does principled optimization by using *minimum-norm* updates for W_i. During this learning theory section, we assume stable dynamics, which we investigate in more detail in Section 4.

3.1 DFC dynamically inverts the output error

To understand how the weight update (5) can access error information, we start by investigating the steady state of the network dynamics (1) and the controller dynamics (4), assuming that all weights are fixed (hence, a separation of timescales). As the feedback controller controls all layers simultaneously, we introduce a compact notation for: concatenated neuron activations $\mathbf{v} \triangleq [\mathbf{v}_1^T, ..., \mathbf{v}_L^T]^T$, feedforward compartments $\mathbf{v}^{\text{ff}} \triangleq [\mathbf{v}_1^{\text{ff},T}, ..., \mathbf{v}_L^{\text{ff},T}]^T$, and feedback weights $Q \triangleq [Q_1^T ... Q_L^T]^T$. Lemma 1 shows a first-order Taylor approximation of the steady-state solution (full proof in Appendix A.1).

Lemma 1. *Assuming stable dynamics, a small target stepsize λ, and W_i and Q_i fixed, the steady-state solutions of the dynamical systems* (1) *and* (4) *can be approximated by*

$$\mathbf{u}_{\text{ss}} = (JQ + \tilde{\alpha} I)^{-1} \boldsymbol{\delta}_L + \mathcal{O}(\lambda^2), \quad \mathbf{v}_{\text{ss}} = \mathbf{v}_{\text{ss}}^{\text{ff}} + Q(JQ + \tilde{\alpha} I)^{-1} \boldsymbol{\delta}_L + \mathcal{O}(\lambda^2), \tag{6}$$

with $J \triangleq \frac{\partial \mathbf{r}_L^-}{\partial \mathbf{v}}\Big|_{\mathbf{v}=\mathbf{v}^-}$ the Jacobian of the network output w.r.t. $\mathbf{v}$, evaluated at the network equilibrium without feedback, $\boldsymbol{\delta}_L$ the output error as defined in (3)*, $\mathbf{v}_{i,ss}^{\text{ff}} = W_i \phi(\mathbf{v}_{i-1,\text{ss}})$, and $\tilde{\alpha} = \alpha/(1 + \alpha k_p)$.*

To get a better intuition of what this steady state represents, consider the scenario where we want to nudge the network activation $\mathbf{v}$ with $\Delta\mathbf{v}$, i.e. $\mathbf{v}_{\text{ss}} = \mathbf{v}_{\text{ss}}^{\text{ff}} + \Delta\mathbf{v}$, such that the steady-state network output equals its target $\mathbf{r}_L^*$. With linearized network dynamics, this results in solving the linear system $J\Delta\mathbf{v} = \boldsymbol{\delta}_L$. As $\Delta\mathbf{v}$ is of much higher dimension than $\boldsymbol{\delta}_L$, this is an underdetermined system with infinitely many solutions. Constraining the solution to the column space of Q leads to the unique solution $\Delta\mathbf{v} = Q(JQ)^{-1}\boldsymbol{\delta}_L$, corresponding to the steady-state solution in Lemma 1 minus a small damping constant $\tilde{\alpha}$. Hence, similar to Podlaski and Machens [31], the controller *dynamically inverts* the output error $\boldsymbol{\delta}_L$ to produce feedback that exactly drives the network output to its desired target.

3.2 DFC approximates Gauss-Newton optimization

To understand the optimization characteristics of DFC, we show that under flexible conditions on Q_i and the layer activations, DFC approximates GN optimization. We first briefly review GN optimization and introduce two conditions needed for the main theorem.

Gauss-Newton optimization [34] is an approximate second-order optimization method used in nonlinear least-squares regression. The GN update for the model parameters $\boldsymbol{\theta}$ is computed as:

$$\Delta\boldsymbol{\theta} = -J_\theta^\dagger \mathbf{e}_L, \tag{7}$$

with J_θ the Jacobian of the model output w.r.t. $\boldsymbol{\theta}$ concatenated for all minibatch samples, $J_\theta^\dagger$ its Moore-Penrose pseudoinverse, and $\mathbf{e}_L$ the output errors.

Condition 1. *Each layer of the network, except from the output layer, has the same activation norm:*

$$\|\mathbf{r}_0\|_2 = \|\mathbf{r}_1\|_2 = ... \|\mathbf{r}_{L-1}\|_2 \triangleq \|\mathbf{r}\|_2. \tag{8}$$

Note that the latter condition considers a statistic $\|\mathbf{r}_i\|_2$ of a whole layer and does not impose specific constraints on single neural firing rates. This condition can be interpreted as each layer, except the output layer, having the same 'energy budget' for firing.

Condition 2. *The column space of Q is equal to the row space of J.*

This more abstract condition imposes a flexible constraint on the feedback weights Q_i, that generalizes common learning rules with direct feedback connections [16, 20]. For instance, besides $Q = J^T$ (BP; [16]) and $Q = J^\dagger$ [20], many other instances of Q which have not yet been explored in the literature fulfill Condition 2 (see Fig. 2), hence leading to principled optimization (see Theorem 2). With these conditions in place, we are ready to state the main theorem of this section (full proof in App. A).

Theorem 2. *Assuming Conditions 1 and 2 hold, J is full rank, the task loss $\mathcal{L}$ is a L^2 loss, and $\lambda, \alpha \to 0$, then the following steady-state (ss) updates for the forward weights,*

$$\Delta W_{i,\text{ss}} = \eta(\mathbf{v}_{i,\text{ss}} - \mathbf{v}^{\text{ff}}_{i,\text{ss}})\mathbf{r}^T_{i-1,\text{ss}} \ , \tag{9}$$

with η a stepsize parameter, align with the weight updates for W_i for the feedforward network (2) *prescribed by the GN optimization method with a minibatch-size of 1.*

In this theorem, we need Condition 2 such that the dynamical inversion $Q(JQ)^{-1}$ (6) equals the pseudoinverse of J and we need Condition 1 to extend this pseudoinverse to the Jacobian of the output w.r.t. the network weights, as in equation (7). Theorem 2 links the DFC method to GN optimization, thereby showing that it does principled optimization, while being fundamentally different from BP. In contrast to recent work that connects target propagation to GN [20, 21], we do not need to approximate the GN curvature matrix by a block-diagonal matrix but use the full curvature instead. Hence, one can use Theorem 2 in Cai et al. [35] to obtain convergence results for this setting of GN with a minibatch size of 1, in highly overparameterized networks. Strikingly, the feedback path of DFC does not need to align with the forward path or its inverse to provide optimally aligned weight updates with GN, as long as it satisfies the flexible Condition 2 (see Fig. 2).

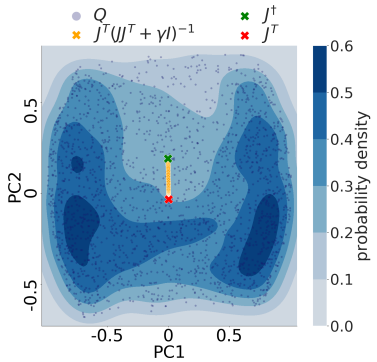

Figure 2: Randomly generated feedback matrices Q (blue) that satisfy Conditions 2 and 3, and have unity norm, visualized by a principal component analysis, with density contours added for visual clarity. J^T, $J^\dagger$, and $J^T(JJ^T + \gamma I)^{-1}$, $\gamma \in [10^{-5}, 10^2]$, are added, highlighting that the optimal feedback configurations for DFC (blue) span a much wider space.

The steady-state updates (9) used in Theorem 2 differ from the actual updates (5) in two nuanced ways. First, the plasticity rule (5) uses a nonlinearity, ϕ, of the compartment activations, whereas in Theorem 2 this nonlinearity is not included. There are two reasons for this: (i) the use of ϕ in (5) can be linked to specific biophysical mechanisms in the pyramidal cell [36] (see Discussion), and (ii) using ϕ makes sure that saturated neurons do not update their forward weights, which leads to better performance (see App. A.5). Second, in Theorem 2, the weights are only updated at steady state, whereas in (5) they are continuously updated during the dynamics of the network and controller. Before settling rapidly, the dynamics oscillate around the steady-state value (see Fig. 1D), and hence, the accumulated continuous updates (5) will be approximately equal to its steady-state equivalent, since the oscillations approximately cancel each other out and the steady-state is quickly reached (see Section 6.1 and App. A.6). Theorem 2 needs a L^2 loss function and Condition 1 and 2 to hold for linking DFC with GN. In the following subsection, we relax these assumptions and show that DFC still does principled optimization.

3.3 DFC uses minimum-norm updates

GN optimization with a minibatch size of 1 is intimately connected to minimum-norm updates [20]. The following theorem shows the connection between DFC and minimum-norm updates, while removing the need for an L^2 loss and Condition 1 (full proof in App. A).

Theorem 3. *Assuming stable dynamics, Condition 2 holds and $\lambda, \alpha \to 0$, the steady-state weight updates* (9) *are proportional to the weighted minimum-norm updates of W_i for letting the feedforward output $\mathbf{r}^-_L$ reach $\mathbf{r}^*_L$, i.e., the solution to the following optimization problem:*

$$\underset{\Delta W_i, i\in[1,..,L]}{\arg\min} \quad \sum_{i=1}^{L} \|\mathbf{r}^{-(m)}_{i-1}\|^2_2 \|\Delta W_i\|^2_F \qquad \textit{s.t.} \quad \mathbf{r}^{-(m+1)}_L = \mathbf{r}^{*(m)}_L, \tag{10}$$

with m the iteration and $\mathbf{r}^{-(m+1)}_L$ the network output without feedback after the weight update.

Theorem 3 shows that Condition 2 enables the controller to drive the network towards its target $\mathbf{r}^*_L$ with minimum-norm activation changes, $\Delta\mathbf{v} = \mathbf{v} - \mathbf{v}^{\text{ff}}$, which combined with the steady-state weight update (9) result in weighted minimum-norm updates ΔW_i (see also App. A.4). When the feedback weights do not have the correct column space, the weight updates will not be minimum norm. Nevertheless, the following proposition shows that the weight updates still follow a descent direction given arbitrary feedback weights.

Proposition 4. *Assuming stable dynamics and $\lambda, \alpha \to 0$, the steady-state weight updates* (9) *with a layer-specific learning rate $\eta_i = \eta / \|r_{i-1}\|_2^2$ lie within 90 degrees of the loss gradient direction.*

4 Stability of DFC

Until now, we assumed that the network dynamics are stable, which is necessary for DFC, as an unstable network will diverge, making learning impossible. In this section, we investigate the conditions on the feedback weights Q_i necessary for stability. To gain intuition, we linearize the network around its feedforward values, assume a separation of timescales between the controller and the network ($\tau_u \gg \tau_v$), and only consider integrative control ($k_p = 0$). This results in the following dynamics (see App. B for the derivation):

$$\tau_u \frac{\mathrm{d}}{\mathrm{d}t} \mathbf{u}(t) = -(JQ + \alpha I)\mathbf{u}(t) + \delta_L. \tag{11}$$

Hence, in this simplified case, the local stability of the network around the equilibrium point depends on the eigenvalues of JQ, which is formalized in the following condition and proposition.

Condition 3. *Given the network Jacobian evaluated at the steady state, $J_{ss} \triangleq \frac{\partial \mathbf{r}_L}{\partial \mathbf{v}}\big|_{\mathbf{v}=\mathbf{v}_{ss}}$, the real parts of the eigenvalues of $J_{ss}Q$ are all greater than $-\alpha$.*

Proposition 5. *Assuming $\tau_u \gg \tau_v$ and $k_p = 0$, the network and controller dynamics are locally asymptotically stable around its equilibrium iff Condition 3 holds.*

This proposition follows directly from Lyapunov's Indirect Method [37]. When assuming the more general case where τ_v is not negligible and $k_p > 0$, the stability criteria quickly become less interpretable (see App. B). However, experimentally, we see that Condition 3 is a good proxy condition for guaranteeing stability in the general case where τ_v is not negligible and $k_p > 0$ (see Section 6 and App. B).

5 Learning the feedback weights

Condition 2 and 3 emphasize the importance of the feedback weights for enabling efficient learning and ensuring stability of the network dynamics, respectively. As the forward weights, and hence the network Jacobian, J, change during training, the set of feedback configurations that satisfy Conditions 2 and 3 also change. This creates the need to adapt the feedback weights accordingly to ensure efficient learning and network stability. We solve this challenge by learning the feedback weights, such that they can adapt to the changing network during training. We separate forward and feedback weight training in alternating wake-sleep phases [38]. Note that in practice, a fast alternation between the two phases is not required (see Section 6).

Inspired by the Weight Mirror method [14], we learn the feedback weights by inserting independent zero-mean noise $\boldsymbol{\epsilon}$ in the system dynamics (1): $\tau_v \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{v}_i(t) = -\mathbf{v}_i(t) + W_i\phi\big(\mathbf{v}_{i-1}(t)\big) + Q_i\mathbf{u}(t) + \sigma\boldsymbol{\epsilon}_i$. The noise fluctuations propagated to the output carry information from the network Jacobian, J. To let $\mathbf{e}$, and hence $\mathbf{u}$, incorporate this noise information, we set the output target $\mathbf{r}_L^*$ to the average network output $\mathbf{r}_L^-$. As the network is continuously perturbed by noise, the controller will try to counteract the noise and regulate the network towards the output target $\mathbf{r}_L^-$. The feedback weights can then be trained with a simple anti-Hebbian plasticity rule with weight decay, which is local in space and time:

$$\tau_Q \frac{\mathrm{d}}{\mathrm{d}t} Q_i(t) = -\mathbf{v}_i^{\mathrm{fb}}(t)\mathbf{u}(t)^T - \beta Q_i, \tag{12}$$

where β is the scale factor of the weight decay term and where we assume that a subset of the noise input $\boldsymbol{\epsilon}_i$ enters through the feedback compartment, i.e., $\mathbf{v}_i^{\mathrm{fb}} = Q_i\mathbf{u} + \sigma_{\mathrm{fb}}\boldsymbol{\epsilon}_i^{\mathrm{fb}}$. The correlation between the noise in $\mathbf{v}_i^{\mathrm{fb}}$ and noise fluctuations in $\mathbf{u}$ provides the teaching signal for Q_i. Theorem 6 shows under simplifying assumptions that the feedback learning rule (12) drives Q_i to satisfy Condition 2 and 3 (see App. C for the full theorem and its proof).

Theorem 6 (Short version)**.** *Assume a separation of timescales $\tau_v \ll \tau_u \ll \tau_Q$, α big, $k_p = 0$, $\mathbf{r}_L^* = \mathbf{r}_L^-$, and Condition 3 holds. Then, for a fixed input sample and $\sigma \to 0$, the first moment of Q converges aproximately to:*

$$\lim_{\sigma \to 0} \mathbb{E}[Q_{ss}] \underset{\sim}{\propto} J^T(JJ^T + \gamma I)^{-1}, \tag{13}$$

for some $\gamma > 0$. Furthermore, $\mathbb{E}[Q_{ss}]$ satisfies Conditions 2 and 3, even if $\alpha = 0$ in the latter.

Theorem 6 shows that under simplifying assumptions, Q converges towards a damped pseudoinverse of J, which satisfies Conditions 2 and 3. Empirically, we see that this also approximately holds for more general settings where τ_v is not negligible, $k_p > 0$, and small α (see Section 6 and App. C).

The above theorem leaves two questions unanswered. First, it assumes that Condition 3 holds, however, the task of the feedback weight training is to make unstable network dynamics stable, resulting in a chicken-and-egg problem. The solution we use is to take α big enough to make the network stable during early training, after which the feedback weights align according to (13) and α can be decreased. Second, Theorem 6 considers training the feedback weights to convergence over one fixed input sample. However, in reality many different input samples will be considered during learning. When the network is linear, J is the same for each input sample and equation (13) holds exactly. However, for nonlinear networks, J will be different for each sample, causing the feedback weights to align with an average of $J^T(JJ^T + \gamma I)^{-1}$ over many samples.

6 Experiments

We evaluate DFC in detail on toy experiments to showcase that our theoretical results translate to practice (Section 6.1) and on a modest range of computer vision benchmarks – MNIST classification and autoencoding [39], and Fashion MNIST classification [40] – to show that DFC can do precise credit assignment in more challenging settings (Section 6.2). Alongside DFC, we test two variants: (i) **DFC-SS** which only updates its feedforward weights W_i after the steady state (SS) of (1) and (4) is reached; and (ii) **DFC-SSA** which analytically computes the steady state of (1) and (4) according to Lemma 1. To investigate whether learning the feedback weights is crucial for DFC, we compare for all three settings: (i) learning the feedback weights Q_i according to (12); and (ii) fixing the feedback weights to the initialization $Q_i = \prod_{k=i+1}^{L} W_k^T$, which approximately satisfies Condition 2 and 3 at the beginning of training (see App. E), denoted with suffix **(fixed)**. For the former, we pre-train the feedback weights according to (12) to ensure stability. During training, we iterate between 1 epoch of forward weight training and X epochs of feedback weight training (if applicable), where $X \in [1, 2, 3]$ is a hyperparameter. We compare all variants to Direct Feedback Alignment (**DFA**) [41] as a control for direct feedback connectivity. DFC is simulated with the Euler-Maruyama method, which is the equivalent of forward Euler for stochastic differential equations [42]. We initialize the network to its feedforward activations (2) for each datasample and, for computational efficiency, we buffer the weight updates (5) and (12) and apply them once at the end of the simulation for the considered datasample. App. D and E provide further details on the implementation of all experiments.[1]

6.1 Toy regression

Figure 3 visualizes the theoretical results of Theorems 2 and 3 and Conditions 1, 2 and 3, in an empirical setting of nonlinear student teacher regression, where a randomly initialized teacher network generates synthetic training data for a student network. We see that Condition 2 is approximately satisfied for all DFC variants that learn their feedback weights (Fig. 3A), leading to close alignment with the ideal minimum norm (MN) updates of Theorem 3 (Fig. 3B). For nonlinear networks and linear direct feedback, it is in general not possible to perfectly satisfy Condition 2 as the network Jacobian J varies for each datasample, while Q_i remains the same. However, the results indicate that feedback learning finds a configuration for Q_i that approximately satisfies Condition 2 for all datasamples. When the feedback weights are fixed, we see that Condition 2 is approximately satisfied in the beginning of training due to a good initialization. However, as the network changes during training, Condition 2 degrades modestly, which results in worse alignment (Fig. 3B).

For having GN updates, both Conditions 1 and 2 need to be satisfied. Although we do not enforce Condition 1 during training, we see in Fig. 3C that it is approximately satisfied, which can be explained by the saturating properties of the $\tanh$ nonlinearity. This is reflected in the alignment with the ideal GN updates in Fig. 3D that follows the same trend as the alignment with the MN updates. Fig. 3E shows that all DFC variants remain stable throughout training, even when the feedback weights are fixed. In App. B, we indicate that Condition 3 is a good proxy for the stability shown in Fig. 3E. Finally, we see in Fig. 3F that the weight updates of DFC and DFC-SS align well with the analytical steady-state solution of Lemma 1, confirming that our learning theory of Section 3 applies to the continuous weight updates (5) of DFC.

[1] PyTorch implementation of all methods is available in the supplementary materials.

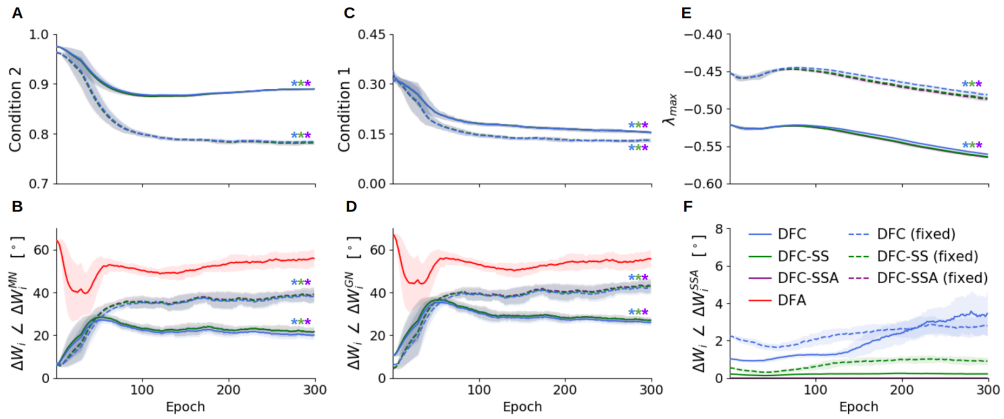

Figure 3: Results for nonlinear student-teacher regression task with layer sizes (15-10-10-5), $\tanh$ nonlinearities, a linear output layer, $k_p = 1.5$, and $\alpha = 0.0015$. (**A**) Ratio between the norms of Q projected into the column space of J^T, and Q, with a value of 1 indicating perfect compliance of Condition 2. (**B**,**D**,**F**) Angle in degrees between the concatenated parameter updates of the whole network and (**B**) the ideal damped minimum-norm parameter updates (i.e., DFC-SSA with Condition 2 exactly satisfied); (**D**) the ideal damped GN parameter updates; and (**F**) the DFC-SSA parameter updates (see App. E.1 for all definitions). (**C**) The standard deviation of the layer norms $\|\mathbf{r}_i\|_2$, divided by the average layer norm, with a value of zero indicating perfect compliance to Condition 1. (**E**) The maximum real part of the eigenvalues of the total system dynamics matrix evaluated at equilibrium (see App. E.1), with negative real parts indicating local stability. For all measures, a window-average is plotted together with the window-std (shade). Stars indicate overlapping plots.

6.2 Computer vision benchmarks

The classification results on MNIST and Fashion-MNIST (Table 1) show that the performances of DFC and its variants, but also its controls, lie close to the performance of BP, indicating that they perform proper credit assignment in these tasks. To see significant differences between the methods, we consider the more challenging task of training an autoencoder on MNIST, where it is known that DFA fails to provide precise credit assignment [9, 16, 31]. The results in Table 1 show that the DFC variants with trained feedback weights clearly outperform DFA and have close performance to BP. The low performance of the DFC variants with fixed feedback weights show the importance of learning the feedback weights continuously during training to satisfy Condition 2. Finally, to disentangle optimization performance from implicit regularization mechanisms, which both influence the test performance, we investigate the performance of all methods in minimizing the training loss of MNIST.[2] The results in Table 1 show improved performance of the DFC method with trained feedback weights compared to BP and controls, suggesting that the approximate minimum-norm updates of DFC can faster descend the loss landscape for this simple dataset.

7 Discussion

We introduced DFC as an alternative biologically-plausible learning method for deep neural networks. DFC uses error feedback to drive the network activations to a desired output target. This process generates a neuron-specific learning signal which can be used to learn both forward and feedback weights locally in time and space. In contrast to other recent methods that learn the feedback weights and aim to approximate BP [14, 15, 16, 17, 25], we show that DFC approximates Gauss-Newton (GN) optimization, making it fundamentally different from BP approximations.

DFC is optimal – i.e., Conditions 2 and 3 are satisfied – for a wide range of feedback connectivity strengths. Thus, we prove that principled learning can be achieved with local rules and without symmetric feedforward and feedback connectivity. This finding has interesting implications for experimental neuroscientific research looking for precise patterns of symmetric connectivity in the brain. Moreover, from a computational standpoint, the flexibility that stems from Conditions 2 and 3 might be relevant for other mechanisms besides learning, such as attention and prediction [8].

[2] We used separate hyperparameter configurations, selected for minimizing the training loss.

Table 1: Test errors (classification) and test loss (autoencoder) corresponding to the epoch with the best validation result (for 5000 validation samples) over a training of 100 epochs (classification) or 25 epochs (autoencoder). Training loss after 100 epochs (MNIST train loss). We use the Adam optimizer [43]. Architectures: 3x256 fully connected (FC) tanh hidden layers and softmax output (classification), 256-32-256 FC hidden layers for autoencoder MNIST with tanh-linear-tanh nonlinearities, and a linear output. Mean $\pm$ std (5 random seeds). Best results (except BP) are displayed in bold.

	MNIST	Fashion-MNIST	MNIST autoencoder	MNIST (train loss)
BP	$2.08^{\pm 0.15}\%$	$10.60^{\pm 0.34}\%$	$9.42^{\pm 0.09} \cdot 10^{-2}$	$1.53^{\pm 0.19} \cdot 10^{-7}$
DFC	$2.25^{\pm 0.094}\%$	$11.17^{\pm 0.27}\%$	$11.28^{\pm 0.18} \cdot 10^{-2}$	$7.61^{\pm 0.65} \cdot 10^{-8}$
DFC-SSA	$\mathbf{2.18^{\pm 0.16}}\%$	$11.28^{\pm 0.27}\%$	$11.27^{\pm 0.09} \cdot 10^{-2}$	$4.89^{\pm 1.26} \cdot 10^{-8}$
DFC-SS	$2.29^{\pm 0.097}\%$	$\mathbf{11.15^{\pm 0.32}}\%$	$\mathbf{11.21^{\pm 0.04} \cdot 10^{-2}}$	$\mathbf{4.80^{\pm 0.70} \cdot 10^{-8}}$
DFC (fixed)	$2.47^{\pm 0.12}\%$	$11.62^{\pm 0.30}\%$	$33.37^{\pm 0.60} \cdot 10^{-2}$	$1.30^{\pm 0.15} \cdot 10^{-6}$
DFC-SSA (fixed)	$2.46^{\pm 0.11}\%$	$11.44^{\pm 0.14}\%$	$31.90^{\pm 0.77} \cdot 10^{-2}$	$1.73^{\pm 0.39} \cdot 10^{-6}$
DFC-SS (fixed)	$2.39^{\pm 0.22}\%$	$11.55^{\pm 0.42}\%$	$32.31^{\pm 0.37} \cdot 10^{-2}$	$1.67^{\pm 0.70} \cdot 10^{-6}$
DFA	$2.69^{\pm 0.11}\%$	$11.38^{\pm 0.25}\%$	$29.95^{\pm 0.36} \cdot 10^{-2}$	$7.09^{\pm 1.11} \cdot 10^{-7}$

To present DFC in its simplest form, we used direct feedback mappings from the output controller to all hidden layers. Although numerous anatomical studies of the mammalian neocortex reported the occurrence of such direct feedback connections [44, 45], it is unlikely that all feedback pathways are direct. We note that DFC is also compatible with other feedback mappings, such as layerwise connections or separate feedback pathways with multiple layers of neurons (see App. G).

Interestingly, the three-compartment neuron is closely linked to recent multi-compartment models of the cortical pyramidal neuron [22, 24, 25, 46]. In the terminology of these models, our central, feedforward, and feedback compartments, correspond to the somatic, basal dendritic, and apical dendritic compartments of pyramidal neurons, respectively (see Fig. 1C). In line with DFC, experimental observations [47, 48] suggest that feedforward connections converge onto the basal compartment and feedback connections onto the apical compartment. Moreover, our plasticity rule for the forward weights (5) belongs to a class of dendritic predictive plasticity rules for which a biological implementation based on backpropagating action potentials has been put forward [36].

Limitations and future work. In practice, the forward weight updates are not exactly equal to GN or MN updates (Theorems 2 and 3), due to limited training iterations for the feedback weights and the limited capacity of linear feedback mappings to satisfy Condition 2 for each datasample. Figure 3 shows that DFC approximates the theory well, however, future work can improve the results further by investigating new feedback architectures (see App. G). We note that GN optimization has desirable approximate second-order optimization properties, however, it is presently unclear whether these second-order characteristics translate to our setting with a minibatch size of 1. Currently, our proposed feedback learning rule (12) aims to approximate one specific configuration and hence does not capitalize on the increased flexibility of DFC and Condition 2. Therefore, an interesting future direction is to design more flexible feedback learning rules that aim to satisfy Conditions 2 and 3 without targeting one specific configuration. Furthermore, DFC needs two separate phases for training the forward weights and feedback weights. Interestingly, if the feedback plasticity rule (12) uses a high-passed filtered version of the presynaptic input $\mathbf{u}$, both phases can be merged into one, with plasticity always on for both forward and feedback weights (see App. C.3). Finally, as DFC is dynamical in nature, it is costly to simulate on commonly used hardware for deep learning, prohibiting us from testing DFC on large-scale problems such as those considered by Bartunov et al. [10]. A promising alternative is to implement DFC on analog hardware, where the dynamics of DFC can correspond to real physical processes on a chip. This would not only make DFC resource-efficient, but also position DFC as an interesting training method for analog implementations of deep neural networks, commonly used in Edge AI and other applications where low energy consumption is key [49, 50].

To conclude, we show that DFC can provide principled credit assignment in deep neural networks by actively using error feedback to drive neural activations. The flexible requirements for feedback mappings combined with the strong link between DFC and GN, underline that it is possible to do principled credit assignment in neural networks without adhering to the symmetric layer-wise feedback structure imposed by BP.

References

[1] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[2] Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.

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

[4] Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

[5] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, 1987.

[6] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, pages 1–12, 2020.

[7] Matthew E Larkum, Thomas Nevian, Maya Sandler, Alon Polsky, and Jackie Schiller. Synaptic integration in tuft dendrites of layer 5 pyramidal neurons: a new unifying principle. *Science*, 325(5941):756–760, 2009.

[8] Charles D Gilbert and Wu Li. Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14(5):350–363, 2013.

[9] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, 2016.

[10] Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems 31*, pages 9368–9378, 2018.

[11] Julien Launay, Iacopo Poli, and Florent Krzakala. Principled training of neural networks with direct feedback alignment. *arXiv preprint arXiv:1906.04554*, 2019.

[12] Theodore H Moskovitz, Ashok Litwin-Kumar, and LF Abbott. Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488*, 2018.

[13] Brian Alexander Crafton, Abhinav Parihar, Evan Gebhardt, and Arijit Raychowdhury. Direct feedback alignment with sparse connections for local learning. *Frontiers in Neuroscience*, 13: 525, 2019.

[14] Mohamed Akrout, Collin Wilson, Peter Humphreys, Timothy Lillicrap, and Douglas B Tweed. Deep learning without weight transport. In *Advances in Neural Information Processing Systems 32*, pages 974–982, 2019.

[15] Daniel Kunin, Aran Nayebi, Javier Sagastuy-Brena, Surya Ganguli, Jonathan Bloom, and Daniel Yamins. Two routes to scalable credit assignment without weight symmetry. In *International Conference on Machine Learning*, pages 5511–5521. PMLR, 2020.

[16] Benjamin James Lansdell, Prashanth Prakash, and Konrad Paul Kording. Learning to solve the credit assignment problem. In *International Conference on Learning Representations*, 2020.

[17] Jordan Guerguiev, Konrad Kording, and Blake Richards. Spike-based causal inference for weight alignment. In *International Conference on Learning Representations*, 2020.

[18] Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*, 2014.

[19] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.

[20] Alexander Meulemans, Francesco Carzaniga, Johan Suykens, João Sacramento, and Benjamin F. Grewe. A theoretical framework for target propagation. *Advances in Neural Information Processing Systems*, 33:20024–20036, 2020.

[21] Yoshua Bengio. Deriving differential target propagation from iterating approximate inverses. *arXiv preprint arXiv:2007.15139*, 2020.

[22] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems 31*, pages 8721–8732, 2018.

[23] James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

[24] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *ELife*, 6:e22901, 2017.

[25] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature neuroscience*, 24(5):1546, 2021.

[26] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[27] Aditya Gilra and Wulfram Gerstner. Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network. *Elife*, 6:e28295, 2017.

[28] Sophie Denève, Alireza Alemi, and Ralph Bourdoukan. The brain as an efficient and robust adaptive learner. *Neuron*, 94(5):969–977, 2017.

[29] Alireza Alemi, Christian Machens, Sophie Denève, and Jean-Jacques Slotine. Learning arbitrary dynamics in efficient, balanced spiking networks using local plasticity rules. *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[30] Ralph Bourdoukan and Sophie Deneve. Enforcing balance allows local supervised learning in spiking recurrent networks. *Advances in Neural Information Processing Systems*, 28:982–990, 2015.

[31] William F Podlaski and Christian K Machens. Biological credit assignment through dynamic inversion of feedforward networks. *Advances in Neural Information Processing Systems 33*, 2020.

[32] Adam A Kohan, Edward A Rietman, and Hava T Siegelmann. Error forward-propagation: Reusing feedforward connections to propagate errors in deep learning. *arXiv preprint arXiv:1808.03357*, 2018.

[33] Gene F Franklin, J David Powell, and Abbas Emami-Naeini. *Feedback control of dynamic systems*. Pearson London, 2015.

[34] Carl Friedrich Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*, volume 7. Perthes et Besser, 1809.

[35] Tianle Cai, Ruiqi Gao, Jikai Hou, Siyu Chen, Dong Wang, Di He, Zhihua Zhang, and Liwei Wang. A gram-gauss-newton method learning overparameterized deep neural networks for regression problems. *arXiv preprint arXiv:1905.11675*, 2019.

[36] Robert Urbanczik and Walter Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528, 2014.

[37] A. M. Lyapunov. The general problem of the stability of motion. *International Journal of Control*, 55(3):531–534, 1992. doi: 10.1080/00207179208934253.

[38] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[39] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[40] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[41] Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in neural information processing systems*, pages 1037–1045, 2016.

[42] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2014.

[44] Leslie G Ungerleider, Thelma W Galkin, Robert Desimone, and Ricardo Gattass. Cortical connections of area v4 in the macaque. *Cerebral Cortex*, 18(3):477–499, 2008.

[45] Kathleen S Rockland and Gary W Van Hoesen. Direct temporal-occipital feedback connections to striate cortex (v1) in the macaque monkey. *Cerebral cortex*, 4(3):300–313, 1994.

[46] Blake A Richards and Timothy P Lillicrap. Dendritic solutions to the credit assignment problem. *Current opinion in neurobiology*, 54:28–36, 2019.

[47] Matthew Larkum. A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex. *Trends in neurosciences*, 36(3):141–151, 2013.

[48] Nelson Spruston. Pyramidal neurons: dendritic structure and synaptic integration. *Nature Reviews Neuroscience*, 9(3):206–221, 2008.

[49] T Patrick Xiao, Christopher H Bennett, Ben Feinberg, Sapan Agarwal, and Matthew J Marinella. Analog architectures for neural network acceleration based on non-volatile memory. *Applied Physics Reviews*, 7(3):031301, 2020.

[50] Janardan Misra and Indranil Saha. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, 74(1-3):239–255, 2010.

[51] Eliakim H Moore. On the reciprocal of the general algebraic matrix. *Bull. Am. Math. Soc.*, 26: 394–395, 1920.

[52] Roger Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.

[53] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2):164–168, 1944.

[54] Nicol N Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural computation*, 14(7):1723–1738, 2002.

[55] William Fulton. Eigenvalues, invariant factors, highest weights, and schubert calculus. *Bulletin of the American Mathematical Society*, 37(3):209–249, 2000.

[56] D Bejarano, Eduardo Ibargüen-Mondragón, and Enith Amanda Gómez-Hernández. A stability test for non linear systems of ordinary differential equations based on the gershgorin circles. *Contemporary Engineering Sciences*, 11(91):4541–4548, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See 'Limitations and future work' paragraph in the Discussion.
   (c) Did you discuss any potential negative societal impacts of your work? [No] As our work is of theoretical nature and considers training feedforward neural networks in general, without focusing on any specific practical application area, we do not anticipate any direct societal impact originating from this work. On the long-term, our work can have impact on related research communities such as neuroscience and deep learning, with societal impact depending on the development of these fields.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We stated in each Theorem, Proposition or Lemma the assumptions needed for proving it.

(b) Did you include complete proofs of all theoretical results? [Yes] See Appendix sections A, B and C. Furthermore, we provided short intuitive pointers for the proofs of the main results, e.g. line 131 and 166-170.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Supplemental Material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendices D and E
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 1
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.6
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite the creators of the datasets we use in line 273 and we cite Pytorch in the appendices.
   (b) Did you mention the license of the assets? [Yes] See Appendix E.7
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We included the code repository that we created to run all experiments in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]