A cookbook for hardware-friendly implicit learning on static data

Maxence Ernoult[♦]
maxence@rain.ai

Rasmus Kjær Høier [*]
hier@chalmers.se

Jack Kendall[♦]
jack@rain.ai

Abstract

The following aims to be a pragmatic introduction to hardware-friendly learning of *implicit models*, which encompass a broad class of models from feedforward nets to physical systems, taking static data as inputs. Starting from first principles, we present a *minimal* hierarchy of independent concepts to circumvent some problems inherent to the hardware implementation of standard differentiation. This way, we avoid entangling essential ingredients with arbitrary design choices by naively listing existing algorithms and instead propose the draft of a "cookbook" to help the exploration of many possible combinations of these independent mechanisms.

1 Problem statement

Learning at equilibrium. Given an input x, we want to find a set of model parameters θ which minimizes a given objective $\mathcal{O}$ defined over the model (hidden and output) variables $s(x,\theta)$, such that the model variables abides by some constraint $\mathcal{C}$. Namely:

$$\mathcal{P}_1: \quad \min_\theta \ J(x,\theta) := \mathcal{O}(s(x,\theta),\theta) \quad \text{s.t.} \quad \mathcal{C}(x, s(x,\theta),\theta) = 0. \tag{1}$$

Note that s may implicitly contain several layers, e.g. $s = ({s^1}^\top, {s^2}^\top, \cdots, {s^N}^\top)^\top$. Classically, $\mathcal{O}$ is some *cost function* ℓ measuring the discrepancy between the model prediction and some ground-truth data and $\mathcal{C}$ the "physical laws" that governs the substrate sustaining the model prediction *at equilibrium* – note that this implicit formalism encompasses both feedforward models (65), deep equilibrium models (4), as well as resistive networks governed by Kirchoff's laws (42; 96; 85). The problem defined in Eq. (1) is traditionally solved via first-order optimization by estimating the gradient over a minibatch $\mathcal{B}$ (106): $\overline{g}(\theta) := \mathbb{E}_{x\sim\mathcal{D}}[d_\theta J(x,\theta)]$. Therefore, solving $\mathcal{P}_1$ boils down to *how to compute* these gradients. We herein call a "circuit" the abstract physical system that may realize a given equilibrium condition.

The Lagrangian method. One way to solve $\mathcal{P}_1$ is by writing the associated *Lagrangian* functional $\mathcal{L}_1(s,\lambda,\theta) := \mathcal{O}(s,\theta) + \lambda^\top \cdot \mathcal{C}(s,\theta)$ and solving for the *Karush-Kuhn-Tucker* (KKT) conditions (89; 13). Loosely speaking, the *primal* feasibility condition $\partial_\lambda \mathcal{L}_1 = 0$ and *dual* feasibility condition $\partial_s \mathcal{L}_1 = 0$ generalize the notions of "forward pass" and "backward pass" respectively, and can be viewed as two circuits determining s and λ whose equilibrium satisfy:

$$\begin{cases} \mathcal{C}(s,\theta) = 0, \\ \partial_s \mathcal{C}(s,\theta)^\top \cdot \lambda + \nabla_s \mathcal{O}(s,\theta) = 0, \end{cases} \tag{2}$$

with the resulting gradient estimate reading as $g(\theta) = \nabla_\theta \mathcal{O}(s,\theta) + \partial_\theta \mathcal{C}(s,\theta)^\top \cdot \lambda$. We briefly mention that this can also be derived using the *Implicit Function Theorem* (IFT) (106). Here as well,

[*]Chalmers University of Technology, Sweden
[♦]Rain AI

note that *feedforward nets trained by backprop are a special case of Eq. (2).* Yet, it appears from Eq. (2) that naively solving for s and λ comes with some challenges from the (high-level) hardware viewpoint: i) **the transport problem**: s and θ need to be transported from the inference circuit to the error circuit; ii) **the memory problem**: s needs to be stored; iii) **analytical derivatives**: $\partial_s \mathcal{C}$ potentially contains analytical derivatives which may have to be computed with high precision; iv) **forward locking**: in a hierarchical model, the circuit computing s^k becomes idle when computing s^{k+1} (Eq. (3)); v) **backward locking**: symmetrically, λ_k cannot be solved before λ_{k+1} (Eq. (6)); vi) **forward–backward synchrony**: λ is computed *after* s.

2 Algorithm design & general methods

2.1 Crafting the constraints

Hierarchical constraints. As the variable s may subsume a hierarchy of layers, one may want to split the original optimization problem into a hierarchy of K problems, given $s := ({s^1}^\top, \cdots, {s^K}^\top)^\top$ (16; 51; 31; 105; 75):

$$\mathcal{P}_2 : \ \min_\theta \ J(x,\theta) := \mathcal{O}(s,\theta) \quad \text{s.t.} \quad \mathcal{C}_1(s^1, s^0 := x, \theta^1) = 0, \cdots, \mathcal{C}_K(s^K, s^{K-1}, \theta^K) = 0 \quad (3)$$

Here, the model is split into a hierarchy of K subcircuits (or "blocks") with parameters θ^k, state s^k, which may comprise one *or multiple layers* ($K \leq L$), subject to the influence of the previous circuit through s^{k-1}. Note that for a given set of constraints, $\mathcal{P}_1$ *is generally not equivalent to* $\mathcal{P}_2$ – see § 3.6 for an example. In this case, solving for the KKT conditions of $\mathcal{P}_2$ yields:

$$\mathcal{C}_k(s^k, s^{k-1}, \theta^k) = 0 \quad (4)$$

$$\partial_{s^K} \mathcal{C}_K(s^K, s^{K-1}, \theta^K)^\top \cdot \lambda^K + \nabla_{s^K} \mathcal{O}(s,\theta) = 0 \quad (5)$$

$$\partial_{s^k} \mathcal{C}_k(s^k, s^{k-1}, \theta^k)^\top \cdot \lambda^k + \nabla_{s^k} \mathcal{O}(s,\theta) + \partial_s^k \mathcal{C}_{k+1}(s^{k+1}, s^k, \theta^{k+1})^\top \cdot \lambda^{k+1} = 0. \quad (6)$$

Note that $\mathcal{O}(s,\theta) = \ell(s^K, y)$ corresponds to the classical "end-to-end" supervised learning setting in which case the second term of Eq. (6) vanishes except for the last block s^K.

Relaxed constraints. When $\mathcal{C} = \nabla_s \mathcal{K}$ (see Section 3), $\mathcal{P}_1$ can be relaxed through an optimal value reformulation with a *fixed* Lagrangian multiplier β^{-1} as (79; 105; 37):

$$\mathcal{P}_3 : \ \min_{\theta,s} \mathcal{O}(s,\theta) \text{ s.t. } \mathcal{K}(s,\theta) \leq \min_{s'} \mathcal{K}(s',\theta), \quad \mathcal{L}_3(s,\theta,\beta) := \mathcal{O}(s,\theta) + \frac{1}{\beta}\left(\mathcal{K}(s,\theta) - \min_{s'} \mathcal{K}(s',\theta)\right) \quad (7)$$

The resulting Lagrangian $\mathcal{L}_3$, sometimes called "surrogate", is intimately tied to energy-based learning (88) (Section 3) and is especially convenient to express quantities as *finite* differences in β which would otherwise appear as *exact* derivatives in β starting from $\mathcal{P}_1$ (which amounts to send $\beta \to 0$). Importantly, this relaxation could also well be applied *at each level of the hierarchy of* $\mathcal{P}_2$ (37).

2.2 Picking an algorithm to estimate λ's: standard approaches

Implicit differentiation. In spite of the aforementioned problems inherent to Eq. (2), one may still directly solve for λ in combination with other tricks (see section 2.3). For feedforward models, $\partial_s \mathcal{C}^\top$ may be *explicitly* inverted with $\lambda = -\partial_s \mathcal{C}^{-\top} \cdot \mathcal{O}$ reducing to *backprop*. In other cases, λ may be computed by *implicit differentiation* (44; 10; 30), which comes in many different flavors depending on the constraints at hand (14). Note that these techniques can be slightly adjusted to accommodate some constraints. For instance, to avoid weight transport between the inference and error circuits, we can equip the error circuit with its own "feedback" parameters ω such that the error signal satisfies: $\partial_s C(s,\omega)^\top \cdot \lambda + \nabla_s \mathcal{O}(s,\omega) = 0$ (52; 76; 1; 48). While randomly sampled ω are sufficient for shallow architectures and simple tasks, some extra alignment mechanisms are needed to have ω roughly approximate θ *without explicitly transporting it* (71; 6; 102; 1; 47).

Another factorization of the gradient. Note that $g(\theta)$ could also well be factorized as $g(\theta) = \nabla_\theta \mathcal{O}(s,\theta) + d_\theta s^\top \cdot \nabla_s \mathcal{O}(s,\theta)$, where $d_\theta s$ satisfies the equilibrium condition:

$$\partial_s \mathcal{C}(s,\theta) \cdot d_\theta s + \partial_\theta \mathcal{C}(s,\theta) = 0. \quad (8)$$

In this case, the error signal is carried *forward* through $\partial_s \mathcal{C}(s, \theta)$, however its memory usage scales cubically with the number of neurons. While low-rank approximations of Eq. (8) have been proposed in the context of RNNs to mitigate its memory cost (101; 98; 11), an extreme way to trade memory for time complexity would be by perturbing every single weight by a small amount ϵ and measure the resulting loss change, yielding:

$$\mathcal{C}(x, s^{\pm\epsilon, e_i}, \theta \pm \epsilon e_i) = 0, \quad d_\theta \mathcal{O}(s, \theta) \approx \sum_{i=1}^{d} \frac{1}{2\epsilon} \left(\mathcal{O}(s^{\epsilon, e_i}, \theta + \epsilon e_i) - \mathcal{O}(s^{-\epsilon, e_i}, \theta - \epsilon e_i) \right) e_i, \tag{9}$$

where $\{e_i\}_{i=1\cdots n}$ denote the canonical basis in the weight space. However, Eq. (9) is still highly inefficient as it takes $2d$ forward passes per iteration to compute a single gradient.

Forward-only learning with zeroth order optimization. One sweet spot between Eq. (8) and Eq. (9), which has recently regained some popularity, is *zeroth order optimization* (ZO) (53). ZO techniques estimate a projection of the gradient $g(\theta)$ along some direction u in the weight space by performing multiple forward passes:

$$u^\top \cdot g(\theta) = d_\epsilon \left(J(\theta + \epsilon u) \right)|_{\epsilon=0} \approx \frac{1}{2\epsilon} \left(\mathcal{O}(s(\theta + \epsilon u), \theta + \epsilon u) - \mathcal{O}(s(\theta - \epsilon u), \theta - \epsilon u) \right), \tag{10}$$

An unbiased estimate of the gradient can be obtained by averaging multiple such derivatives: $g(\theta) = \mathbb{E}_{u \sim \mathcal{N}(0, \sigma^2)} \left[d_\epsilon \left(J(\theta + \epsilon u) \right)|_{\epsilon=0} \cdot u \right]$ (94; 7). However, the variance of this gradient estimate scales cubically with the number of model parameters (80), thereby restricting the applicability of ZO to the realm of models which are small enough (93; 35; 7; 26) or behaving as such, i.e. *pre-trained models* (56). One way to mitigate this problem is to perturb neurons instead of synapses (25; 66; 15; 43), namely computing projections of the error signal as: $v^\top \cdot \lambda = d_\epsilon \mathcal{O}(s^\epsilon, \theta)|_{\epsilon=0}$ with s^ϵ implicitly determined through $\mathcal{C}(s^\epsilon, \theta) + \epsilon v = 0$. Similarly, an unbiased gradient estimate is obtained as $g(\theta) = \nabla_\theta \mathcal{O} + \partial_\theta \mathcal{C}(s, \theta)^\top \cdot \mathbb{E}_{v \sim \mathcal{N}(0, \sigma^2)} \left[d_\epsilon \mathcal{O}(s^\epsilon, \theta)|_{\epsilon=0} \cdot v \right]$. Going beyond, one can teach small auxiliary networks synthetizing such directions to output "good" weight directions u instead of randomly sampling them (26), reducing the variance of the gradient estimate at the cost of increasing the bias (81; 93).

2.3 Other tricks which independently apply

Greedy learning & loss design. It appears from Eq. (6) that in general, an error signal λ^{k+1} must be passed backwards to "unlock" the computation of λ^k. In order to parallelize learning across blocks entirely, a heuristic consists in shutting off the top-down error signal ($\lambda^{k+1} = 0$) and recreate an error signal through a *locally-defined* (supervised (8; 77; 9; 99; 29) or self-supervised (55; 103; 33; 92)) loss, which amounts to choosing $\mathcal{O} = \mathcal{O}_1(s^1, \theta^1) + \cdots + \mathcal{O}_K(s^K, \theta^K)$. In this case, the resulting block satisfies the exact same adjoint equation as that of the original learning problem (Eq. (2)) so that all gradient computation techniques herein presented may also apply *block-wise*. Alternatively, another solution to backward locking is to *estimate* λ_{k+1} with auxiliary modules (41).

Checkpointing & reversible models. One way to mitigate the memory problem mentioned above (at the expense of compute) is to *simultaneously* compute s and λ, which can be viewed as activation checkpointing (18). In models with an explicit layer hierarchy ($\mathcal{P}_2$), another special instantiation of activation checkpointing is possibly when using *reversible models* (21; 28; 17; 57): a given constraint $\mathcal{C}_{k+1}(s^{k+1}, s^k) = 0$ can be explicitely inverted as $\mathcal{C}_{k+1}^{-1}(s^k, s^{k+1}) = 0$ such that s^k can be recomputed *backward* from layer s^{k+1} instead of being stored, or recomputed *forward* from $s^0 = x$. This can be achieved for instance by splitting each block state s^k into two, $s^k = ({s_a^k}^\top, {s_b^k}^\top)^\top$, and defining $\mathcal{C}_{k+1} = (\mathcal{C}_{k+1}^a, \mathcal{C}_{k+1}^b)$, with dedicated transformations f_a, f_b, as (28):

$$\mathcal{C}_{k+1}^a(s^{k+1}, s^k) = s_a^{k+1} - s_a^k - f_a(s_b^k, \theta_a^k), \quad \mathcal{C}_b^{k+1}(s^{k+1}, s^k) = s_b^{k+1} - s_b^k - f_b(s_a^k, \theta_b^k) \tag{11}$$

Pipelining. When dealing with a hierarchical model ($\mathcal{P}_2$), the block $\mathcal{C}_k$ may become *idle* or "forward-locked" after passing s^k to $\mathcal{C}_{k+1}$ until next input comes in. A solution to this is to push *multiple inputs in sequence* through the blocks, allowing them to process different inputs *in parallel*, e.g. $\mathcal{C}_k$ processes input x_{p+1} while $\mathcal{C}_{k+1}$ processes input x_p, the same strategy applying *backwards* for the computation of the λ_k's for different inputs (40). As naive pipelining may still maintain idleness "bubbles", more elaborate schemes have been proposed (24), for instance by allowing each block to alternate between forward pass and backward passes for different inputs (73), yet at the cost of *gradient staleness* – λ is computed with a different θ that the one used to compute s. This problem can be mitigated for instances by by maintaining different weight versions at each circuit (74).

3 Forward-only learning beyond zeroth order

3.1 Energy-based (EB) models & Energy-based Learning (EBL)

When the constraint of the optimization problem Eq. (1) derives from an energy function $\mathcal{K}$, energy-based learning (EBL) refers to a family of gradient computation algorithms which implicitly estimate s and λ using a *single* circuit (39; 34; 72; 5; 96; 86; 88). Namely, if there exists some scalar function $\mathcal{K}$ such that $\mathcal{C} := \nabla_s \mathcal{K}$, defining $\mathcal{F} := \mathcal{K} + \beta \mathcal{O}$ where $\beta \geq 0$ is some scalar value, then one can estimate $g(\theta)$ by having the *same* circuit relax twice to equilibrium with two different values of β and subsequent "nudged states" $s_{\pm\beta}$(86; 46):

$$s_{\pm\beta} : \nabla_s \mathcal{F}(\pm\beta, s_{\pm\beta}, \theta) = 0, \quad g(\theta) = \frac{1}{2\beta}\left(\nabla_\theta \mathcal{F}(\beta, s_\beta, \theta) - \nabla_\theta \mathcal{F}(-\beta, s_{-\beta}, \theta)\right) + \mathcal{O}(\beta^2) \quad (12)$$

Eq. (12) is in stark contrast with Eq. (2), though being mathematically equivalent through $\lambda = d_\beta s_\beta|_{\beta=0}$ (86; 75): instead of estimating s and λ on two circuits, s_β and $s_{-\beta}$ are estimated on a *single* circuit through "energy minimization", which is why $\mathcal{C}$ is said to be *energy based* (EB). The core intuition behind the magic of EB learning is that while error signals are usually carried "backward" through $\partial_s \mathcal{C}^\top$ (Eq. (2)), since we have $\partial_s \mathcal{C}^\top = \nabla_s^2 \mathcal{K} = \partial_s \mathcal{C}$, error signals can in this case be equivalently carried "forward" through $\partial_s \mathcal{C}$ through a small perturbation of s along $\nabla_s \mathcal{O}$ of sufficiently small β. Finally note that Eq. (12) is only one many variants (88) to estimate $d_\beta(\nabla_\theta \mathcal{F})|_{\beta=0}$ (86), where there is trade-off between the number of s_β being evaluated and the resulting bias (107; 45).

3.2 The importance of nudging

The weak & strong nudging limits. The most current EBL setting is when $\mathcal{C}$ describes the implicit model itself and $\mathcal{O}$ the cost function: $(\mathcal{C}, \mathcal{O}) = (F = \nabla_s E, \ell)$ where we have denoted $K = E$. In this case, the condition on the nudged state reads (86):

$$\nabla_s E(s_\beta, \theta) + \beta \nabla_s \ell(s_\beta, \theta) = 0 \quad (13)$$

We call the *nudging* factor the scalar controlling the strength of the cost function ℓ in the definition of s_β. For this choice of $(\mathcal{C}, \mathcal{O})$, β is the nudging factor and since theoretical guarantees of Eq. (12) hold for $\beta \to 0$, it corresponds to small $\nabla_s \ell$ perturbations, which is therefore called the *weak nudging limit*. Conversely, when swapping the objective and the constraint (64; 65), i.e. $(\mathcal{C}, \mathcal{O}) = (\nabla_s \ell, E)$, which amounts to swap E and ℓ inside Eq. (13), one can see that β^{-1} now gates the error signal $\nabla_s \ell$. Therefore in this situation, $\beta \to 0$ corresponds to *large* $\nabla_s \ell$ perturbations, which is therefore called the *strong nudging limit*. While it seems counter-intuitive that strong nudging solutions are relevant – the main goal of of Eq. (1) should be to minimize the loss subject to constraints on the network energy and not the other way around – they are global minimizers of $\theta \to \ell(\theta, s(\theta))$ for certain choices of E (65). Strong nudging can enable learning in situations where the physical system at use is noisy by improving the signal to noise ratio for the teaching signal (64; 45).

Nudging through adiabatic oscillations. EBL classically operates in the weak nudging regime and estimates the error signal $\lambda = d_\beta s_\beta|_{\beta=0}$ through a discrete, two-steps finite difference procedure, as highlighted in Eq. (12). Another way to view the error signal is as the contour integration in the complex plane $\lambda = \frac{1}{T|\beta|}\int_0^T e^{i2\pi t/T} s_{\beta(t)}$ with $\beta(t) := |\beta| e^{i2\pi/T}$. Then, provided T is sufficiently large compared to the characteristic time constant which governs the relaxation of $\mathcal{C}$ to equilibrium, the error signal can be computed through slow adiabatic oscillations of the system (5; 45; 3).

Nudging heuristics. The error signal $\beta \nabla_s \ell$ may be transmitted *instantaneously* to the rest of the network, or with a delay. Denoting $h(\theta, s)$ the model logits, some output controller dynamics u may integrate the error signal inside the output layer as $\tau_u \dot{u} + u = -\beta \nabla_o \ell(h(s, \theta))$ and feed it back to the rest of the circuit (65). There also exists other types of nudging among the broader family of

Contrastive Learning (CL) algorithms. Denoting y some label, the "nudged" hidden state h_y can be defined by hard-clamping y to the output units, $\nabla_s E(h_y, o = y, \theta) = 0$, while prescribing a weight update of the same contrastive form as Eq. 12 (72). An hybrid between weak nudging and target clamping consists of clamping the output to a *weakly nudged target*, resulting in a nudged hidden state h^β defined as: $\nabla_s E(h^\beta, o = (1-\beta)o_\star + \beta y) = 0$ where $o_\star$ corresponds to the free state ($\beta = 0$) of the output (96).

Nudging synapses. One may wonder whether neurons and parameters could play a symmetrical role, by having them *both* evolve throughout the gradient computation phase or satisfy an equilibrium condition. One common *post hoc* solution is to play on characteristic time constants by having parameters slowly integrate the *instantaneous* contrastive updates prescribed by Eq. (12) (22; 97; 45; 20), which however requires the detailed knowledge of the energy function at use. Another solution circumventing this constraint is to let *both* s and θ equilibrate in the same energy landscape, with control variables u acting upon θ with strength α (87). When computing $s_{-\beta}$ (for instance), u is adjusted to keep θ at its current value θ_t such that $u_t = \theta_t + \alpha \nabla_s \mathcal{F}(-\beta, s_{-\beta}, \theta_t)$. Therefore, keeping u_t constant when computing s_β, θ equilibrates at: $\theta_{t+1} = u_t - \alpha \nabla_s \mathcal{F}(\beta, s_\beta, \theta_{t+1}) \approx \theta_t - \alpha\beta g(\theta)$.

3.3 Two phases or twice as many neurons?

Lagrangian reparametrization. One may usually regard the contrastive update Eq. (12) as a procedure performed on the *same* physical system whose state is measured at two different times with two different nudging values (e.g. $-\beta$, β), or a continuum of such. Yet, another view of Eq. (12) is to assume that the two nudged states are computed *simultaneoulsy by two different circuits* sharing the same parameters. An advantage of this implementation, which has been experimentally demonstrated with transistor-based synapses (20), is that it requires a single phase only instead of two, yet at the expense of area and complex engineering to enable parameter sharing. A different way to approach the same question is through the concept of *dyadic neurons* (36; 37): two variables s_+ and s_- are such that their (convex) sum encodes the model prediction and their difference the error signal if they are critical points of the following reparametrized Lagrangian (37):

$$\widetilde{\mathcal{L}}_1(s_+, s_-, \theta) := \mathcal{L}_1\left(\alpha s_+ + (1-\alpha)s_-, (s_+ - s_-)/\beta\right), \tag{14}$$

where $\mathcal{L}$ defined inside Eq. (2) and $\alpha \in [0, 1]$. While the resulting gradient computation algorithm is generally a reparametrization of implicit differentiation by construction, when the constraint at use inside Eq. (1) is energy based and β is sufficiently small, $s_\pm$ can be simply construed as $s_{\pm\beta}$ of Eq. (12). This can be shown by performing this reparametrization on the *relaxed* Lagrangian $\mathcal{L}_3$ (Eq. (7)) with *fixed* β (37), or equivalently approximated from Eq. (14) in the limit $\beta \to 0$.

3.4 Applying EBL to implicit models

Weak nudging. One of the ways to cast an implicit model F, which does not explicitly derive from an energy function, into an EB model is simply to employ the energy function $\mathcal{K} = E = \frac{1}{2}\|F\|^2$ (65). An important case, as one means to train feedforward nets by EP, is when $F(s, \theta) = s - f(\theta, s)$ where θ is typically lower block diagonal (27; 100; 67; 69). Eq. (12) for some β, in the weak nudging regime ($(\mathcal{O}, \mathcal{K}) = (\ell, E)$), yields in this case:

$$\begin{cases} s_\beta = f(s_\beta, \theta) + \partial_s f(s_\beta, \theta)^\top \cdot (s_\beta - f(s_\beta, \theta)) - \beta \nabla_s \ell(s_\beta, \theta), \\ g(\theta) = \frac{1}{\beta} \partial_\theta f(s_\beta, \theta)^\top \cdot (s_\beta - f(s_\beta, \theta)) + \mathcal{O}(\beta) \end{cases} \tag{15}$$

A nice feature of Eq. (15) is that by construction, $s_\beta = f(s_\beta, \theta)$ when $\beta = 0$ such that the gradient can be estimated up to $\mathcal{O}(\beta)$ with a *single* nudged state. However, the presence of $\partial_s f(s_\beta, \theta)^\top$ signals the potential use analytical derivatives. To obviate this, one may instead estimate the required derivatives through finite differences of *feedback operators* (50; 82; 62; 63; 23; 61) as: $s_\beta \approx f(s_\beta, \theta) + g(s_\beta, \omega) - g(f(s_\beta), \omega)) - \beta \nabla_s \ell(s_\beta, \theta)$ where the feedback parameters may have to be learned so that $\partial_s g \approx \partial_s f^\top$ (23). However a problem remaining is that both s_β and their "feedforward" prediction $f(s_\beta)$ need to be simultaneously maintained to implement Eq. (15). A solution is to use auxiliary variables ψ which track the feedforward activity such that $\psi_\beta \approx f_\psi(s_\beta, \theta)$ and $s_\beta \approx f(s_\beta, \theta) + g(s_\beta, \omega) - g(\psi_\beta, \omega) - \beta \nabla_s \ell(s_\beta, \theta)$ (82).

Recovering a least control problem in the strong nudging regime. When in the strong nudging regime, the optimization problem Eq. (1) using $E = \frac{1}{2}\|F\|^2 = \mathcal{O}$ along with $\ell = \mathcal{K}$ acquires a very intriguing meaning, though the loss is not picked as the objective. Defining $\psi_\beta := -F(s_\beta, \theta)$, the learning problem amounts to find the set of parameters which minimizes the amount of *control* $\mathcal{O} = \frac{1}{2}\|\psi_\beta\|^2$ such that the controlled equilibrium $\psi_\beta + F(s_\beta, \theta) = 0$ minimizes the loss ℓ, which works in practice on feedforward and implicit models more broadly (65). However, this interpretation is tied to the choice $E = \frac{1}{2}\|F\|^2$ and further investigation is needed to extend the applicability of strong nudging and its connection to control theory beyond this choice of energy function.

3.5 Handling nonlinearity inside EB models

When not handled cautiously, the application of Eq. 12 to implicit models yields activation derivatives (Eq. (15)). While neuroscientists, seeking for biologically plausible learning theories, gave birth to the aforementioned approximations of Eq. (15) and many others, these remain impractical for hardware-friendly learning. One *ad hoc* solution used in practice, when the inverse of the activation function is continuously invertible, is to separate the linear and nonlinear contribution inside the energy function as $E = G + U$ where G is defined such that $\nabla_s G = \sigma^{-1}$ (39; 105; 36; 88). A more principled solution to handle σ is to instead treat it as a *constraint* $\mathcal{C}_\sigma$ on the set of feasible configurations over which energy minimization is performed, which descent steps are *hard* projected onto(85). A "relaxed" version of this approach, somewhat bridging with the former solution, is through the lens of *mirror descent* using the Bregman divergence associated with G (2): $s_{t+1} = \arg\min\{s^\top \cdot \nabla_s E(s) + \frac{1}{\eta}\Delta_G(s, s_t)\}$ with $\Delta_G(x, y) := G(x) - G(y) + \nabla G(y)^\top \cdot (x - y)$. Note that many other fixed point algorithms could be use for the same purpose (14).

3.6 Casting feedforward nets into hierarchical energy-based models

An important example. Another way to cast feedforward nets training by EBL algorithms is to break them into multiple, hierarchical energy-based constraints based on Eq. (3) (105). The following example, which shows how to do so, is also a good illustration of why the problems $\mathcal{P}_1$ (Eq. (1)) and $\mathcal{P}_2$ (Eq. (3)) are generally not equivalent. Taking $E := \sum_{k=1}^{K} E_k$ with $E_k(s^k, s^{k-1}, \theta^k) := G(s^k) - s^{k^\top} \cdot \theta^k \cdot s^{k-1}$, we consider $\mathcal{P}_1$ with $\mathcal{C} = \nabla_s E$ and $\mathcal{P}_2$ with $\mathcal{C}_k = \nabla_{s^k} E^k$. In this case, s^k *is a single layer* and not a block of several layers. Setting the constraint to zero in these two cases yields:

$$\mathcal{P}_1 : \; s^k = \sigma\left(\theta^k \cdot s^{k-1} + \theta^{k+1^\top} \cdot s^{k+1}\right), \quad \mathcal{P}_2 : \; s^k = \sigma\left(\theta^k \cdot s^{k-1}\right) \tag{16}$$

In one case, the model retained at inference time is a *Hopfield model* and in the other one a *feedforward* model. This difference is instrumental to understand why works revolving around "contrastive learning" may actually apply to feedforward models through the $\mathcal{P}_2$ problem formulation (36). Likewise, the nudged state defined by Eq. (12) reads for any layer *until penultimate* the same as Eq. (16) for problem $\mathcal{P}_1$ and as follows for $\mathcal{P}_2$, taking β sufficiently small (105; 75):

$$\mathcal{P}_2 : \; s^k_{\pm\beta} = \sigma\left(\theta^k \cdot s^{k-1} \pm \beta\theta^{k+1^\top} \cdot d_\beta s^{k+1}_\beta|_{\beta=0}\right) \approx \sigma\left(\theta^k \cdot s^{k-1} \pm \theta^{k+1^\top} \cdot \left(s^{k+1}_\beta - s^{k+1}_{-\beta}\right)/2\right), \tag{17}$$

Note that many other choices of energy functions are possible in order to learn feedforward nets as a hierarchical energy-based model (16; 51; 31; 105)

Breaking the forward–backward synchrony. As seen from Eq. (17), computing $s^k_{\pm\beta}$ still requires the storage of s^{k-1}. A trick is to notice that for sufficiently small β, $s^k \approx (s^k_\beta + s^k_{-\beta})/2 := \bar{s}_\beta$ such that $s^k_\beta \approx \sigma\left(\theta^k \cdot \bar{s}_\beta \pm \theta^{k+1^\top} \cdot \Delta s^{k+1}_\beta\right)$ with $\Delta s^{k+1}_\beta := s^{k+1}_\beta - s^{k+1}_{-\beta}$. Implemented this way, the gradient corresponding to problem $\mathcal{P}^2$ with the above choice of energy function can be computed by *simultaneously* solving for s^k_β's and $s^k_{-\beta}$'s (36). This implementation can also be derived by picking hierarchical constraints (Eq. (3)), relaxing these (Eq. (7)) and reparametrizing them (Eq. (14)) (37).

4 Looking ahead & take-aways

Going out of equilibrium? One of the ways to leverage equilibrium techniques towards learning from sequences is to treat the neuron *velocity* $\dot{s}$ as a variable of its own inside the energy function

(32; 90), or to put it differently to pick the *Lagrangian* of the physical system as an energy function such that the resulting equilibrium is an actual trajectory (84). Another exciting route beyond equilibrium models is to consider system which have physical access to their *adjoint* trajectory through *time-reversible*, Hamiltonian-based dynamics (54). This requirement can be regarded as the condition to be self-adjoint in the situation where the constraint $\mathcal{C}$ at use is a *trajectory* rather than an equilibrium state. In such a case, the associated algorithmic baseline is no longer implicit differentiation but the *continuous adjoint state method* (83).

Forward-only learning beyond first order. One may wonder if (and how) *second-order optimization* could be emulated using a *single* circuit. A second-order update generally prescribes $\Delta\theta_{2^{\text{nd}}} \propto \mathbb{E}_{x\sim\mathcal{D}}[C(x,\theta)]^{-1} \cdot \mathbb{E}_{x\sim\mathcal{D}}[g(x,\theta)]$ where $\mathcal{C}$ is a curvature matrix, i.e. the loss *Hessian* or p.s.d. approximations thereof (58). As such, computing $\theta_{2^{\text{nd}}}$ generally subsumes (*Hessian-free* approaches being a notable exception (60)): i) computing g, ii) computing C, iii) inverting C, iv) computing $\Delta\theta_{2^{\text{nd}}}$. However, an interesting insight arises when $|\mathcal{D}| = 1$ (unit batch size), picking the Fisher information curvature matrix $C = (\partial_\theta s^\top \cdot \partial_\theta s)$ (59) and assuming $\mathcal{O}$ does not explicitly depend on θ for simplicity. Then, denoting $A^\dagger$ the *Moore-Penrose pseudo-inverse* of a matrix A:

$$\Delta\theta_{2^{\text{nd}}} \propto \left(\partial_\theta s^\top \cdot \partial_\theta s\right)^{-1} \cdot \partial_\theta s^\top \cdot \nabla_s \mathcal{O} = \partial_\theta s^\dagger \cdot \nabla_s \mathcal{O}. \tag{18}$$

Therefore, moving to second order optimization here amounts to route the error signal $\nabla_s\mathcal{O}$ backward through the system's *inverse* rather than its adjoint. While this viewpoint has motivated several implementations of *self-invertible* (rather than *self-adjoint*, or equivalently energy-based) circuits (50; 62; 12; 63), it may not straightforwardly generalize to the mini-batch regime ($|\mathcal{D}| > 1$).

Finding energy functions that map to physical systems. Given a system described by the implicit mapping $\mathcal{C} = 0$, one can always pick $E = 1/2 \times \|\mathcal{C}\|^2$ (65), or even other variants picking other distances beyond the Euclidean metric, as a valid energy function. Also, the hierarchical energy-based setting depicted above offers even more flexibility to cast many feedforward architectures into an energy-based model. Beyond these tricks, other important computational primitives such as batch normalization or attention can also be construed as energy-based operations (38), yet to be investigated in the context of forward-only energy-based learning. So at first glance, "everything" seems to be energy-based and therefore trainable by the aforementioned forward-only techniques. However, all energy functions may not necessarily map to a physical system, namely a lot of these energy functions may *not* be physical. For instance, the fact that layered resistive networks can compute MLP operations at equilibrium *and* are governed by the minimization of an energy function that is itself the pseudo-power of these circuits (42; 85) is truly remarkable. Therefore, finding an energy function which realizes some core computational primitive *while being physically feasible* is difficult.

Analog-digital codesign. While *fully analog* learning experimental proof-of-concepts have been realized on small systems (20; 104), bringing analog systems at chip scale inherently requires digital circuitry for registers, tile-to-tile routing, analog non idealities mitigation (ADCs & DACs), and more fundamentally for the implementation of analytical and possibly non-weight stationnary operations (95; 49; 104). Beyond this practical aspect, one may wonder if we even *should* map all operations into analog. For instance, attention operations are known to be memory-bounded both at inference and training time when using large context windows rather than bottlenecked by the digital compute itself (91; 19). So it is unclear whether or not analog computing may necessarily bring an advantage for this particular example. Overall this observation calls for digital-analog codesign, and to a further extent for the design of new theoretical building blocks that would account for both digital *and* analog parts of a circuit with associated learning algorithms (75).

Conclusion. This "cookbook" is an attempt to complement recent algorithm unification papers (105; 106; 68; 78; 70) using pragmatic, ingredient-based logics. For instance, "Predictive coding" (100) amounts to apply Eq. (12) with a specific choice of energy function (3.4) and nudging scheme (3.2). Likewise, the "Forward-forward" algorithm (33) is a greedy learning strategy, with each block comprising a single layer, using local self-supervised loss (2.3). Conversely, one may want to split an architecture into blocks (Eq. (3)), apply standard implicit differentiation in some (2.2, §1), ZO in others (2.2, §1), along with a pipelining mechanism (2.3) depending on the constraints at hand, which does not correspond to any known algorithm in the literature.

References

[1] M. Akrout, C. Wilson, P. Humphreys, T. Lillicrap, and D. B. Tweed. Deep learning without weight transport. *Advances in neural information processing systems*, 32, 2019.

[2] E. Amid, R. Anil, and M. Warmuth. Locoprop: Enhancing backprop via local loss optimization. In *International Conference on Artificial Intelligence and Statistics*, pages 9626–9642. PMLR, 2022.

[3] V. R. Anisetti, A. Kandala, B. Scellier, and J. Schwarz. Frequency propagation: Multi-mechanism learning in nonlinear physical networks. *Neural Computation*, 36(4):596–620, 2024.

[4] S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. *Advances in neural information processing systems*, 32, 2019.

[5] P. Baldi and F. Pineda. Contrastive learning and neural oscillations. *Neural computation*, 3(4): 526–545, 1991.

[6] S. Bartunov, A. Santoro, B. Richards, L. Marris, G. E. Hinton, and T. Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31, 2018.

[7] A. G. Baydin, B. A. Pearlmutter, D. Syme, F. Wood, and P. Torr. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

[8] E. Belilovsky, M. Eickenberg, and E. Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pages 583–593. PMLR, 2019.

[9] E. Belilovsky, L. Leconte, L. Caccia, M. Eickenberg, and E. Oyallon. Decoupled greedy learning of cnns for synchronous and asynchronous distributed learning. *arXiv preprint arXiv:2106.06401*, 2021.

[10] B. M. Bell and J. V. Burke. Algorithmic differentiation of implicit functions and optimal values. In *Advances in automatic differentiation*, pages 67–77. Springer, 2008.

[11] G. Bellec, F. Scherr, E. Hajek, D. Salaj, R. Legenstein, and W. Maass. Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv preprint arXiv:1901.09049*, 2019.

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

[13] D. P. Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[14] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert. Efficient and modular implicit differentiation. *Advances in neural information processing systems*, 35:5230–5242, 2022.

[15] G. Bouvier, J. Aljadeff, C. Clopath, C. Bimbard, J. Ranft, A. Blot, J.-P. Nadal, N. Brunel, V. Hakim, and B. Barbour. Cerebellar learning using perturbations. *Elife*, 7:e31599, 2018.

[16] M. Carreira-Perpinan and W. Wang. Distributed optimization of deeply nested systems. In *Artificial Intelligence and Statistics*, pages 10–19. PMLR, 2014.

[17] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[18] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

[19] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35: 16344–16359, 2022.

[20] S. Dillavou, B. D. Beyer, M. Stern, M. Z. Miskin, A. J. Liu, and D. J. Durian. Machine learning without a processor: Emergent learning in a nonlinear electronic metamaterial. *arXiv preprint arXiv:2311.00537*, 2023.

[21] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[22] M. Ernoult, J. Grollier, D. Querlioz, Y. Bengio, and B. Scellier. Equilibrium propagation with continual weight updates. *arXiv preprint arXiv:2005.04168*, 2020.

[23] M. M. Ernoult, F. Normandin, A. Moudgil, S. Spinney, E. Belilovsky, I. Rish, B. Richards, and Y. Bengio. Towards scaling difference target propagation by learning backprop targets. In *International Conference on Machine Learning*, pages 5968–5987. PMLR, 2022.

[24] S. Fan, Y. Rong, C. Meng, Z. Cao, S. Wang, Z. Zheng, C. Wu, G. Long, J. Yang, L. Xia, et al. Dapple: A pipelined data parallel approach for training large models. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 431–445, 2021.

[25] I. R. Fiete, M. S. Fee, and H. S. Seung. Model of birdsong learning based on gradient estimation by dynamic perturbation of neural conductances. *Journal of neurophysiology*, 98 (4):2038–2057, 2007.

[26] L. Fournier, S. Rivaud, E. Belilovsky, M. Eickenberg, and E. Oyallon. Can forward gradient match backpropagation? In *International Conference on Machine Learning*, pages 10249–10264. PMLR, 2023.

[27] K. Friston and S. Kiebel. Predictive coding under the free-energy principle. *Philosophical transactions of the Royal Society B: Biological sciences*, 364(1521):1211–1221, 2009.

[28] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017.

[29] A. N. Gomez, O. Key, K. Perlin, S. Gou, N. Frosst, J. Dean, and Y. Gal. Interlocking backpropagation: Improving depthwise model-parallelism. *Journal of Machine Learning Research*, 23(171):1–28, 2022.

[30] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[31] F. Gu, A. Askari, and L. El Ghaoui. Fenchel lifted networks: A lagrange relaxation of neural network training. In *International Conference on Artificial Intelligence and Statistics*, pages 3362–3371. PMLR, 2020.

[32] P. Haider, B. Ellenberger, L. Kriener, J. Jordan, W. Senn, and M. A. Petrovici. Latent equilibrium: A unified learning theory for arbitrarily fast computation with arbitrarily slow neurons. *Advances in neural information processing systems*, 34:17839–17851, 2021.

[33] G. Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[34] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley. *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984.

[35] N. Hiratani, Y. Mehta, T. Lillicrap, and P. E. Latham. On the stability and scalability of node perturbation learning. *Advances in Neural Information Processing Systems*, 35:31929–31941, 2022.

[36] R. Høier, D. Staudt, and C. Zach. Dual propagation: Accelerating contrastive hebbian learning with dyadic neurons. In *International Conference on Machine Learning*, pages 13141–13156. PMLR, 2023.

[37] R. K. Høier and C. Zach. Two tales of single-phase contrastive hebbian learning. *arXiv preprint arXiv:2402.08573*, 2024.

[38] B. Hoover, Y. Liang, B. Pham, R. Panda, H. Strobelt, D. H. Chau, M. Zaki, and D. Krotov. Energy transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

[39] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

[40] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *Advances in neural information processing systems*, 32, 2019.

[41] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *International conference on machine learning*, pages 1627–1635. PMLR, 2017.

[42] J. Kendall, R. Pantone, K. Manickavasagam, Y. Bengio, and B. Scellier. Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981*, 2020.

[43] J. Kornfeld, M. Januszewski, P. Schubert, V. Jain, W. Denk, and M. S. Fee. An anatomical substrate of credit assignment in reinforcement learning. *BioRxiv*, pages 2020–02, 2020.

[44] S. G. Krantz and H. R. Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

[45] A. Laborieux and F. Zenke. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. *Advances in neural information processing systems*, 35: 12950–12963, 2022.

[46] A. Laborieux, M. Ernoult, B. Scellier, Y. Bengio, J. Grollier, and D. Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:633674, 2021.

[47] B. J. Lansdell, P. R. Prakash, and K. P. Kording. Learning to solve the credit assignment problem. *arXiv preprint arXiv:1906.00889*, 2019.

[48] J. Launay, I. Poli, F. Boniface, and F. Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. *Advances in neural information processing systems*, 33: 9346–9360, 2020.

[49] M. Le Gallo, R. Khaddam-Aljameh, M. Stanisavljevic, A. Vasilopoulos, B. Kersting, M. Dazzi, G. Karunaratne, M. Brändli, A. Singh, S. M. Mueller, et al. A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nature Electronics*, 6(9):680–693, 2023.

[50] D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pages 498–515. Springer, 2015.

[51] J. Li, C. Fang, and Z. Lin. Lifted proximal operator machines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4181–4188, 2019.

[52] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014.

[53] S. Liu, P.-Y. Chen, B. Kailkhura, G. Zhang, A. O. Hero III, and P. K. Varshney. A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54, 2020.

[54] V. Lopez-Pastor and F. Marquardt. Self-learning machines based on hamiltonian echo backpropagation. *Physical Review X*, 13(3):031020, 2023.

[55] S. Löwe, P. O'Connor, and B. Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. *Advances in neural information processing systems*, 32, 2019.

[56] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

[57] K. Mangalam, H. Fan, Y. Li, C.-Y. Wu, B. Xiong, C. Feichtenhofer, and J. Malik. Reversible vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10830–10840, 2022.

[58] J. Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

[59] J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[60] J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1033–1040, 2011.

[61] K. Max, L. Kriener, G. Pineda García, T. Nowotny, I. Jaras, W. Senn, and M. A. Petrovici. Learning efficient backprojections across cortical hierarchies in real time. *Nature Machine Intelligence*, pages 1–12, 2024.

[62] A. Meulemans, F. Carzaniga, J. Suykens, J. Sacramento, and B. F. Grewe. A theoretical framework for target propagation. *Advances in Neural Information Processing Systems*, 33: 20024–20036, 2020.

[63] A. Meulemans, M. Tristany Farinha, J. García Ordóñez, P. Vilimelis Aceituno, J. Sacramento, and B. F. Grewe. Credit assignment in neural networks through deep feedback control. *Advances in Neural Information Processing Systems*, 34:4674–4687, 2021.

[64] A. Meulemans, M. T. Farinha, M. R. Cervera, J. Sacramento, and B. F. Grewe. Minimizing control for credit assignment with strong feedback. In *International Conference on Machine Learning*, pages 15458–15483. PMLR, 2022.

[65] A. Meulemans, N. Zucchet, S. Kobayashi, J. Von Oswald, and J. Sacramento. The least-control principle for local learning at equilibrium. *Advances in Neural Information Processing Systems*, 35:33603–33617, 2022.

[66] T. Miconi. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *Elife*, 6:e20899, 2017.

[67] B. Millidge, A. Seth, and C. L. Buckley. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2021.

[68] B. Millidge, Y. Song, T. Salvatori, T. Lukasiewicz, and R. Bogacz. Backpropagation at the infinitesimal inference limit of energy-based models: Unifying predictive coding, equilibrium propagation, and contrastive hebbian learning. *arXiv preprint arXiv:2206.02629*, 2022.

[69] B. Millidge, A. Tschantz, and C. L. Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368, 2022.

[70] A. Momeni, B. Rahmani, B. Scellier, L. G. Wright, P. L. McMahon, C. C. Wanjura, Y. Li, A. Skalli, N. G. Berloff, T. Onodera, et al. Training of physical neural networks. *arXiv preprint arXiv:2406.03372*, 2024.

[71] T. H. Moskovitz, A. Litwin-Kumar, and L. Abbott. Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488*, 2018.

[72] J. R. Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models*, pages 10–17. Elsevier, 1991.

[73] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia. Pipedream: Generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM symposium on operating systems principles*, pages 1–15, 2019.

[74] D. Narayanan, A. Phanishayee, K. Shi, X. Chen, and M. Zaharia. Memory-efficient pipeline-parallel dnn training. In *International Conference on Machine Learning*, pages 7937–7947. PMLR, 2021.

[75] T. Nest and M. Ernoult. Towards training digitally-tied analog blocks via hybrid gradient computation. *arXiv preprint arXiv:2409.03306*, 2024.

[76] A. Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[77] A. Nøkland and L. H. Eidnes. Training neural networks with local error signals. In *International conference on machine learning*, pages 4839–4850. PMLR, 2019.

[78] A. Ororbia, A. Mali, A. Kohan, B. Millidge, and T. Salvatori. A review of neuroscience-inspired machine learning. *arXiv preprint arXiv:2403.18929*, 2024.

[79] J. V. Outrata. A note on the usage of nondifferentiable exact penalties in some special optimization problems. *Kybernetika*, 24(4):251–258, 1988.

[80] M. Ren, S. Kornblith, R. Liao, and G. Hinton. Scaling forward gradient with local losses. *arXiv preprint arXiv:2210.03310*, 2022.

[81] Y. Ruan, Y. Xiong, S. Reddi, S. Kumar, and C.-J. Hsieh. Learning to learn by zeroth-order oracle. *arXiv preprint arXiv:1910.09464*, 2019.

[82] J. Sacramento, R. Ponte Costa, Y. Bengio, and W. Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

[83] F. Sapienza, J. Bolibar, F. Schäfer, B. Groenke, A. Pal, V. Boussange, P. Heimbach, G. Hooker, F. Pérez, P.-O. Persson, et al. Differentiable programming for differential equations: A review. *arXiv preprint arXiv:2406.09699*, 2024.

[84] B. Scellier. A deep learning theory for neural networks grounded in physics. *arXiv preprint arXiv:2103.09985*, 2021.

[85] B. Scellier. A fast algorithm to simulate nonlinear resistive networks. *arXiv preprint arXiv:2402.11674*, 2024.

[86] B. Scellier and Y. Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[87] B. Scellier, S. Mishra, Y. Bengio, and Y. Ollivier. Agnostic physics-driven deep learning. *arXiv preprint arXiv:2205.15021*, 2022.

[88] B. Scellier, M. Ernoult, J. Kendall, and S. Kumar. Energy-based learning algorithms for analog computing: a comparative study. *Advances in Neural Information Processing Systems*, 36, 2024.

[89] G. Scheithauer. Jorge nocedal and stephen j. wright: Numerical optimization, springer series in operations research, 1999, isbn 0-387-98793-2 in the preface the authors state... our goal in this book is to give a compre-hensive description of the most powerful, state-of-the-art, techniques for solving continuous optimization problems. by presenting the motivating ideas for each.

[90] W. Senn, D. Dold, A. F. Kungl, B. Ellenberger, J. Jordan, Y. Bengio, J. Sacramento, and M. A. Petrovici. A neuronal least-action principle for real-time learning in cortical circuits. *BioRxiv*, pages 2023–03, 2023.

[91] N. Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[92] S. A. Siddiqui, D. Krueger, Y. LeCun, and S. Deny. Blockwise self-supervised learning at scale. *arXiv preprint arXiv:2302.01647*, 2023.

[93] D. Silver, A. Goyal, I. Danihelka, M. Hessel, and H. van Hasselt. Learning by directional gradient descent. In *International Conference on Learning Representations*, 2021.

[94] J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341, 1992.

[95] K. Spoon, H. Tsai, A. Chen, M. J. Rasch, S. Ambrogio, C. Mackin, A. Fasoli, A. M. Friz, P. Narayanan, M. Stanisavljevic, et al. Toward software-equivalent accuracy on transformer-based deep neural networks with analog memory devices. *Frontiers in Computational Neuroscience*, 15:675741, 2021.

[96] M. Stern, D. Hexner, J. W. Rocks, and A. J. Liu. Supervised learning in physical networks: From machine learning to learning machines. *Physical Review X*, 11(2):021045, 2021.

[97] M. Stern, S. Dillavou, M. Z. Miskin, D. J. Durian, and A. J. Liu. Physical learning beyond the quasistatic limit. *Physical Review Research*, 4(2):L022037, 2022.

[98] C. Tallec and Y. Ollivier. Unbiased online recurrent optimization. *arXiv preprint arXiv:1702.05043*, 2017.

[99] Y. Wang, Z. Ni, S. Song, L. Yang, and G. Huang. Revisiting locally supervised learning: an alternative to end-to-end training. *arXiv preprint arXiv:2101.10832*, 2021.

[100] J. C. Whittington and R. 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.

[101] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[102] W. Xiao, H. Chen, Q. Liao, and T. Poggio. Biologically-plausible learning algorithms can scale to large datasets. *arXiv preprint arXiv:1811.03567*, 2018.

[103] Y. Xiong, M. Ren, and R. Urtasun. Loco: Local contrastive representation learning. *Advances in neural information processing systems*, 33:11142–11153, 2020.

[104] S.-i. Yi, J. D. Kendall, R. S. Williams, and S. Kumar. Activity-difference training of deep neural networks using memristor crossbars. *Nature Electronics*, 6(1):45–51, 2023.

[105] C. Zach. Bilevel programs meet deep learning: A unifying view on inference learning methods. *arXiv preprint arXiv:2105.07231*, 2021.

[106] N. Zucchet and J. Sacramento. Beyond backpropagation: bilevel optimization through implicit differentiation and equilibrium propagation. *Neural Computation*, 34(12):2309–2346, 2022.

[107] N. Zucchet, S. Schug, J. Von Oswald, D. Zhao, and J. Sacramento. A contrastive rule for meta-learning. *Advances in neural information processing systems*, 35:25921–25936, 2022.