Holomorphic Equilibrium Propagation Computes Exact Gradients Through Finite Size Oscillations

Axel Laborieux[1] **Friedemann Zenke**[1,2]

{firstname.lastname}@fmi.ch
[1] Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland
[2] Faculty of Natural Sciences, University of Basel, Basel, Switzerland

Abstract

Equilibrium propagation (EP) is an alternative to backpropagation (BP) that allows the training of deep neural networks with local learning rules. It thus provides a compelling framework for training neuromorphic systems and understanding learning in neurobiology. However, EP requires infinitesimal teaching signals, thereby limiting its applicability in noisy physical systems. Moreover, the algorithm requires separate temporal phases and has not been applied to large-scale problems. Here we address these issues by extending EP to holomorphic networks. We show analytically that this extension naturally leads to exact gradients even for finite-amplitude teaching signals. Importantly, the gradient can be computed as the first Fourier coefficient from finite neuronal activity oscillations in continuous time without requiring separate phases. Further, we demonstrate in numerical simulations that our approach permits robust estimation of gradients in the presence of noise and that deeper models benefit from the finite teaching signals. Finally, we establish the first benchmark for EP on the ImageNet 32×32 dataset and show that it matches the performance of an equivalent network trained with BP. Our work provides analytical insights that enable scaling EP to large-scale problems and establishes a formal framework for how oscillations could support learning in biological and neuromorphic systems.

1 Introduction

The backpropagation (BP) of error algorithm [1] underpins the ability of state-of-the-art deep neural networks to learn useful representations from structured data such as speech, vision, and text [2]. BP stands out as the most successful algorithm to solve the credit assignment problem in artificial neural networks [3, 4], which can be defined by the following question: How should a synaptic connection be modified in order to improve the global performance of the network to perform a task, as measured by some objective function? This is a difficult question since individual synapse may have a complicated influence on downstream processing. BP solves credit assignment through the chain rule of differentiation [1]. Although BP is efficiently implemented in software, it is difficult to conceive how BP could plausibly be implemented in biological systems. The problematic aspects are BP's use of symmetric connections and the need for two separate phases: A nonlinear forward pass that propagates neuronal activity and a linear backward pass that carries signed gradient signals [3]. These two types of processing are also inconvenient for training physical neural networks, since both should be handled by the same circuit, and explicitly propagating errors does not harness the device mismatches typical of neuromorphic hardware [5, 6]. Despite its implausibility, representations learned with BP match representations of in-vivo data [7] better than networks trained with purely biologically-motivated learning rules such as STDP [8, 9]. This discrepancy raises the question as to

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

whether and how neural dynamics could implement gradient-based credit assignment, and whether it could be as effective as BP to learn useful representations [3].

Equilibrium propagation (EP) [10] is an alternative algorithm for performing credit assignment in dynamical systems that converge to a fixed point, such as energy-based models [11]. EP also proceeds in two phases: In the first phase the dynamical system is presented with static input data until the units settle into an equilibrium or fixed point. We refer to this state as the free equilibrium. In a second phase, a teaching signal slightly nudges designated output units towards a target value until the dynamics settle into a second equilibrium that is called the nudged equilibrium. EP estimates loss gradients by comparing the neuronal activity between the two equilibria. EP is appealing because the resulting learning rule is spatially local when the energy function consists of two-body interactions, as for instance in continuous Hopfield networks [11]. Furthermore EP provably approximates the true gradient in the limit of vanishing nudging [10]. More generally, the implicit differentiation carried out by EP makes it suitable for meta learning [12], where explicitly backpropagating errors through an inner optimization loop becomes prohibitive due to the high memory requirement of storing intermediate time steps for regular automatic differentiation.

Nevertheless, classic EP [10, 13, 14] has several limitations. First, EP estimates only approach the actual loss gradient in the limit of a vanishing nudging or teaching signal. This requirement makes it impractical for noisy neuromorphic systems where noise can confound small amplitude teaching signals and also unrealistic as a model for learning in the brain where feedback strongly modulates processing. Moreover, the mechanisms by which biological circuits could satisfy the requirement for separate phases remains elusive. Finally, while EP can train deep networks on CIFAR-10 [14], it has remained an open question whether it can be scaled up to larger and more complex tasks [4].

In this article, we show that by extending EP with holomorphic network dynamics it naturally estimates exact gradients for finite teaching signals. Mathematically, the exact gradients are encoded as a Fourier coefficient of adiabatic neural oscillations. This finding suggest a natural way of estimating the gradients online through suitable synaptic filtering operations which dispenses with the need for separate phases, in a similar spirit to Baldi and Pineda [15] for contrastive Hebbian learning [16]. Our main contributions are the following:

- We develop the theory of holomorphic EP (hEP) and prove that this allows computing exact gradients locally at synapses from finite teaching signal amplitudes of adiabatic oscillations.
- We numerically quantify the accuracy of our estimate and show that it outperforms classic EP, especially in the presence of substrate noise and in deep neural networks.
- We demonstrate learning with an always-on oscillating teaching signal, thereby alleviating the need for separated phases.
- Finally, we show that hEP achieves the same performance as BP in deep convolutional neural networks (CNNs) trained on CIFAR-10/100 [17], and ImageNet 32×32 [18].

2 Background and previous work

Equilibrium propagation (EP). EP [10] allows training convergent dynamical systems to optimize a loss function. We denote neuronal unit activity by the vector $\mathbf{s}$, and the learnable parameters such as weights and biases by $\boldsymbol{\theta}$ (Fig. 1a). The system's dynamics are given as the gradient of a scalar energy function $E(\boldsymbol{\theta}, \mathbf{s})$:

$$\frac{\mathrm{d}\mathbf{s}}{\mathrm{d}t} = -\frac{\partial E}{\partial \mathbf{s}}(\boldsymbol{\theta}, \mathbf{s}). \tag{1}$$

As a consequence, EP can train any energy-based models, e.g., Hopfield networks [11] to perform classification [10, 13, 14]. In classic EP, training proceeds in two phases. First, a subset of units are clamped to the input $\mathbf{x}$ and the system goes to a 'free' fixed point denoted by $\mathbf{s}_0^*$. Second, the loss function $\ell(\boldsymbol{\theta}, \mathbf{s}, \mathbf{y})$, with target $\mathbf{y}$, is scaled with a small positive nudging factor β and added to the energy function E which yields the total energy $F(\boldsymbol{\theta}, \mathbf{s}, \beta, \mathbf{y}) := E + \beta\ell(\boldsymbol{\theta}, \mathbf{s}, \mathbf{y})$. This added teaching signal causes the system to reach a second equilibrium $\mathbf{s}_\beta^*$, again by minimizing the total energy F. Although we write that ℓ takes all units $\mathbf{s}$ as argument, in practice typically only output units which encode the target label and thus serve as inputs for teaching signals are considered (Fig. 1b). The learning objective of the system is to optimize the loss function ℓ at the free fixed point,

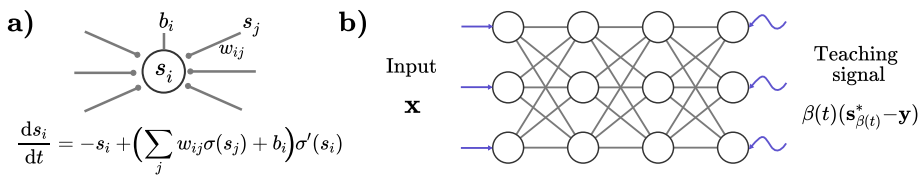

Figure 1: **a)** Schematic of the neuron model of a continuous Hopfield network [11] with holomorphic activation function σ. The neuron s_i receives input both from upstream and downstream neurons s_j plus a bias current b_i. **b)** In a typical supervised learning the input $\mathbf{x}$ is clamped, causing the network dynamics to settle into a fixed point. A complex-valued oscillating teaching signal added to the output causes neuronal activity to fluctuate around this fixed point.

which is defined by $\mathcal{L}(\boldsymbol{\theta}, \mathbf{x}, \mathbf{y}) := \ell(\boldsymbol{\theta}, \mathbf{s}_0^*, \mathbf{y})$. Scellier and Bengio [10] showed that:

$$\lim_{\beta \to 0} \frac{1}{\beta} \left(\frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}_\beta^*, \beta, \mathbf{y}) - \frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}_0^*, 0, \mathbf{y}) \right) = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}}. \tag{2}$$

This result requires F to be twice continuously differentiable and assumes that one can apply the implicit function theorem to the equilibrium equation $\partial_{\mathbf{s}} F(\boldsymbol{\theta}, \mathbf{s}_0^*, 0) = 0$ (the $\mathbf{y}$ argument is hereafter omitted for clarity), so that $\beta \mapsto \mathbf{s}_\beta^*$ is a continuously differentiable map [10]. In practice, the left-hand side of Eq. (2) is estimated by finite differences [10, 13, 14, 19]. The appeal of EP for biological plausibility [3, 20] and neuromorphic hardware [15, 21–23] arises from the fact that (i) the system only needs to propagate neural activities (Fig. 1a) and (ii) in layered neural networks with a Hopfield energy [11] (Fig. 1b), the left-hand side of Eq. (2) can be computed by a Hebbian-like learning rule as the product of pre- and postsynaptic activity. In summary, EP implicitly propagates error signals through differences of neuronal activity during the two phases, whereas BP propagates error gradients explicitly [3, 20]. However, Eq. (2) only holds in the limit $\beta \to 0$ where activity differences vanish, which can pose a problem in the presence of noise or when activity differences vanish with network depth, which is related to the problem of vanishing gradients. In the following, we introduce holomorphic EP (hEP) which avoids these issues by estimating exact gradients with finite β, and thus from finite amplitude teaching signals.

3 Theoretical results

Our main contribution is to show that if F is holomorphic, i.e., differentiable in the sense of complex variables (see Appendix A.1 for the definition), hEP computes the gradient of the objective function $\mathcal{L}$ for finite β, i.e., without requiring the vanishing nudging signals (cf. Eq. (2)). To accomplish this hEP requires a non-vanishing teaching signal that evolves 'adiabatically' in the complex plane with respect to the dynamics of the system (Fig. 1b). In other words, we require the dynamical system to relax to its equilibrium on a much shorter timescale than the timescale of the nudge.

Derivation of holomorphic EP. To show that hEP yields an unbiased gradient through finite adiabatic nudging, we use the same notation as in Section 2. Specifically, we extend the theory by Scellier and Bengio [10] to the complex case and to dynamical systems, or networks whose scalar function governing the dynamics is holomorphic. In line with classic EP we assume that the dynamical system has a free fixed point as described above.

Lemma 1 (Holomorphic Equilibrium Propagation)**.** *Let F be a scalar function governing the dynamics, so that the holomorphic implicit function theorem can be applied to the fixed point equation $\partial_{\mathbf{s}} F(\boldsymbol{\theta}, \mathbf{s}_0^*, 0) = 0$, then the gradient formula of equilibrium propagation (Eq.* (2)*) holds in the sense of complex differentiation.*

Proof. The proof is an extension of the one provided by [10] for the real nudging case. The holomorphic implicit function theorem ensures that there exists an open set $U \in \mathbb{C}$ including 0 such that the implicit map $\beta \in U \mapsto \mathbf{s}_{\boldsymbol{\theta},\beta}^*$ is holomorphic on U. In particular, the fixed point $\mathbf{s}_{\boldsymbol{\theta},\beta}^*$ is defined on U (see Fig. 2b for how this area looks for a toy example). The proof proceeds in two steps. First we show that the total derivatives of F with respect to $\boldsymbol{\theta}$ and β can still be interchanged for complex variables by virtue of the Schwarz theorem. Second we show that, at the fixed point, the

total derivative of F with respect to β ($\boldsymbol{\theta}$) is still equal to the partial derivatives with respect to β ($\boldsymbol{\theta}$). To that end, we apply the chain rule of complex differentiation in:

$$\frac{\mathrm{d}F}{\mathrm{d}\beta}(\boldsymbol{\theta}, \mathbf{s}_{\boldsymbol{\theta},\beta}, \beta) = \frac{\partial F}{\partial \beta}(\boldsymbol{\theta}, \mathbf{s}_{\boldsymbol{\theta},\beta}, \beta) + \frac{\partial F}{\partial \mathbf{s}} \cdot \frac{\partial \mathbf{s}}{\partial \beta}(\boldsymbol{\theta}, \beta) + \frac{\partial F}{\partial \overline{\mathbf{s}}} \cdot \frac{\partial \overline{\mathbf{s}}}{\partial \beta}(\boldsymbol{\theta}, \beta), \tag{3}$$

where $\overline{\mathbf{s}}$ denotes the complex conjugate of $\mathbf{s}$. At equilibrium, the second term on the right hand side cancels by definition of the fixed point, and the third term is zero because F is holomorphic, i.e., its derivative with respect to the conjugate variable is zero according to the Cauchy-Riemann condition [24]. The same argument holds for the derivative with respect to $\boldsymbol{\theta}$. Therefore, interchanging the total derivatives of F with respect to β and $\boldsymbol{\theta}$, and replacing the inner total derivatives by the partial derivatives, we obtain that the EP gradient formula (Eq. (2)) still holds, but for *complex* differentiation (Appendix A.1):

$$\left.\frac{\mathrm{d}}{\mathrm{d}\beta}\right|_{\beta=0} \left(\frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta)\right) = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \frac{\partial F}{\partial \beta}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta) = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}}, \tag{4}$$

which concludes the proof (a more detailed version is in Appendix A.2). □

We can now evaluate the left hand side of Eq. (4) using a Cauchy integral (Appendix A.1):

Theorem 1 (Exact gradient from finite teaching signals)**.** *Assuming that the conditions of Lemma 1 are met and let $|\beta| > 0$ be the radius of a circular path around 0 in $\mathbb{C}$ contained in the open set U on which the fixed point $\mathbf{s}^*_{\boldsymbol{\theta},\beta}$ is defined. Further assume that this path is parameterized by $t \in [0, T] \mapsto \beta(t) = |\beta| e^{2\mathrm{i}\pi t/T}$, where i is the imaginary unit. Then the loss gradient is given by:*

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \frac{1}{T|\beta|} \int_0^T \frac{\partial F}{\partial \boldsymbol{\theta}} \left(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta(t)}, \beta(t)\right) e^{-2\mathrm{i}\pi t/T} \mathrm{d}t. \tag{5}$$

The full proof is given in Appendix A.3. Theorem 1 guarantees that given holomorphic dynamics we can dispense with the requirement of vanishing teaching signal $|\beta| \to 0$ in the limit of 'adiabatic' nudging which corresponds to integrating over infinitely many fixed points with a *finite* $|\beta|$. Note, that complex-valued teaching signals $\beta \in \gamma$ produce fixed points in the complex plane computed through the same equations as in the real case (see Appendix B). In particular, multi-layered neural networks (Fig. 1b) can be trained by using the continuous Hopfield dynamics [11]. The trainable parameters are the weights and biases $\boldsymbol{\theta} = (w_{ij}, b_i)$ and the total energy function F is given by:

$$F(\boldsymbol{\theta}, \mathbf{s}, \beta, \mathbf{y}) = \frac{1}{2}\sum_i s_i^2 - \frac{1}{2}\sum_{i \neq j} w_{i,j}\sigma(s_i)\sigma(s_j) - \sum_i b_i \sigma(s_i) + \beta \ell(\boldsymbol{\theta}, \mathbf{s}, \mathbf{y}). \tag{6}$$

If the activation function σ is holomorphic, which is true in the case of sigmoid functions, the same F (Eq. (6)) can be evaluated with complex β, and we can apply Eq. (5) to obtain:

$$-\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} = \frac{1}{T|\beta|} \int_0^T \sigma_i^*(t)\sigma_j^*(t) e^{-2\mathrm{i}\pi t/T} \mathrm{d}t, \tag{7}$$

where $\sigma_i^*(t) := \sigma(s^*_{i,\beta(t)})$. Therefore, assuming a T-periodic teaching signal (Fig. 1b), the gradient is proportional to the first exponential Fourier coefficient of the product of the oscillating activities. Although this formulation assumes complex neuronal output, we show in Appendix A.4 that the gradient in Eq. (7) can be expressed in terms of the real part or imaginary part only. The complex teaching signal is therefore best thought of as a way to produce unbiased neuronal oscillations on the nudging timescale. In the next section, we numerically estimate the Fourier coefficient of Eq. (5) with a fixed number of points N on the circle which we use to train networks in the subsequent experiments and to compare it to the actual loss gradient computed with automatic differentiation.

Numerical estimation of the loss gradient as a Fourier coefficient. Next, we explain how to estimate the gradient from the corresponding Fourier coefficient (Eq. (5)). In practice, we use a Riemann sum to compute the integral numerically. We fix the nudging radius $|\beta| > 0$ such that the circular path lies in the domain U in which equilibria exist as described in Theorem 1. We sample the path with $N \geq 2$ nudging points $\{\beta_k := |\beta| e^{2\mathrm{i}\pi k/N} \; ; \; k \in [0, ..., N-1]\}$, and define the estimator:

$$\hat{\nabla}(N) := \frac{1}{N|\beta|} \sum_{k=0}^{N-1} \frac{\partial F}{\partial \boldsymbol{\theta}} \left(\boldsymbol{\theta}, \mathbf{s}^*_{\beta_k}, \beta_k\right) e^{-2\mathrm{i}\pi k/N}. \tag{8}$$

We have that $\hat{\nabla}(N) \underset{N\to\infty}{\longrightarrow} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}}$, and the remaining bias term when using N points is:

$$\hat{\nabla}(N) - \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \sum_{p\equiv 0\ (N)}^{\infty} \frac{C_{p+1}|\beta|^p}{(p+1)!}, \tag{9}$$

where C_p is the p-th derivative in 0 of the function $\beta \mapsto \partial_{\boldsymbol{\theta}} F(\boldsymbol{\theta}, \mathbf{s}^*_\beta, \beta)$ (see Appendix A.5 for the proof). The bias term in Eq. 9 converges to zero with increasing N because it is a sub-sum of the $(N+1)$-th order remainder of the series expansion of $\beta \mapsto \partial_{\boldsymbol{\theta}} F(\boldsymbol{\theta}, \mathbf{s}^*_\beta, \beta)$ in 0. The rate of convergence depends on the C_p coefficients and the radius $|\beta|$. In the case $N = 2$, the estimate of Eq. (8) coincides with the 'symmetric' estimate of Laborieux et al. [14]. However, the bias term on the right hand side of Eq. (9) is only valid when the dynamics are holomorphic. Next, we illustrate the approach in a toy experiment, and list three practical improvements brought by hEP.

4 Experiments

In all the experiments, we used the discrete setting of convergent recurrent neural networks of [13], and the readout scheme of [14] for optimizing the cross-entropy loss function with EP (Appendix B). All simulations were implemented in Jax [25] and Haiku [26] (Apache License 2.0). The datasets were obtained from the Tensorflow datasets library [27]. Our code is publicly available on GitHub[1]. The details of simulations and hyperparameters can be found in Appendix E and F.

Demonstration of holomorphic Equilibrium Propagation on a single data point. To provide the first numerical validation of Theorem 1 while also allowing us to gain intuitions about dynamics of individual neurons, we implemented a small-scale multi-layer perceptron (MLP) with layer dimensions 6-4-4-4, including input and output layers. The activation function was a shifted sigmoid, which is holomorphic (Appendix B). The network was fed with a single datapoint, namely a randomly sampled point from a Gaussian and a random one-hot target. The dark blue region in Figure 2b shows the map of complex β for which the network settles to a fixed point after 200 time steps. We found experimentally that the area in the complex plane where stable fixed points exist strongly depends on the activation function and the weight initialization (see Appendix D). As β evolves on the circle of radius 0.1 ($N = 24$) hidden layer neurons settle into different equilibrium points in the complex plane (Fig. 2a). We observed that while the teaching signal $\beta(t) = |\beta|e^{2\mathrm{i}\pi t/T}$ was purely sinusoidal, the non-linearity of the network induces neural oscillations that are not purely sinusoidal. Nevertheless, the gradient is contained in the first mode of these non-linear oscillations (Fig. 2c).

To understand how the gradient is computed accurately when the magnitude of the teaching signal is increased, we recorded the adiabatic product of activities $\sigma^*_i(t)\sigma^*_j(t)$ for one pair of neurons (dark blue) over one teaching period and for increasing values of $|\beta|$ (Fig. 2c). In the case $|\beta| = 0.001$, the perturbation induced by the sinusoidal teaching signal is also purely sinusoidal, and the gradient magnitude is simply the radius of the circle. However, when $|\beta|$ is increased to 0.01, the linear approximation of the perturbation becomes less accurate, because higher powers of β become significant in the series expansion around the free fixed point. The gradient could still be well approximated by taking the mean of the two radii corresponding to real positive and negative β, as done by Laborieux et al. [14]. Increasing $|\beta|$ further to 0.1 and 0.5 yields an even more deformed perturbation, but the gradient is still correctly contained in the first Fourier coefficient of the perturbation. hEP breaks down when β reaches amplitudes for which the corresponding path intersects with areas in which no stable equilibrium exists (cf. Fig. 2b, light areas, and Fig. 2d). However, as we will see in the next sections, the finite teaching amplitudes are beneficial when the neuronal dynamics are subject to noise and when training deep neural networks.

Holomorphic EP can estimate the gradient in continuous time. Our theoretical findings allow us to revisit in a principled way an idea introduced by Baldi and Pineda [15] for a continuous-time-implementation of contrastive Hebbian learning [16] and learning rules looking to maximize slowness [28–30]. In this context, the oscillating teaching signal is always-on, and the dynamical network is governed by three mechanisms acting on distinct timescales. The smallest timescale T_{dyn} is the typical time needed by the network to reach its fixed point. The second timescale is the period of

[1] `https://github.com/Laborieux-Axel/holomorphic_eqprop`

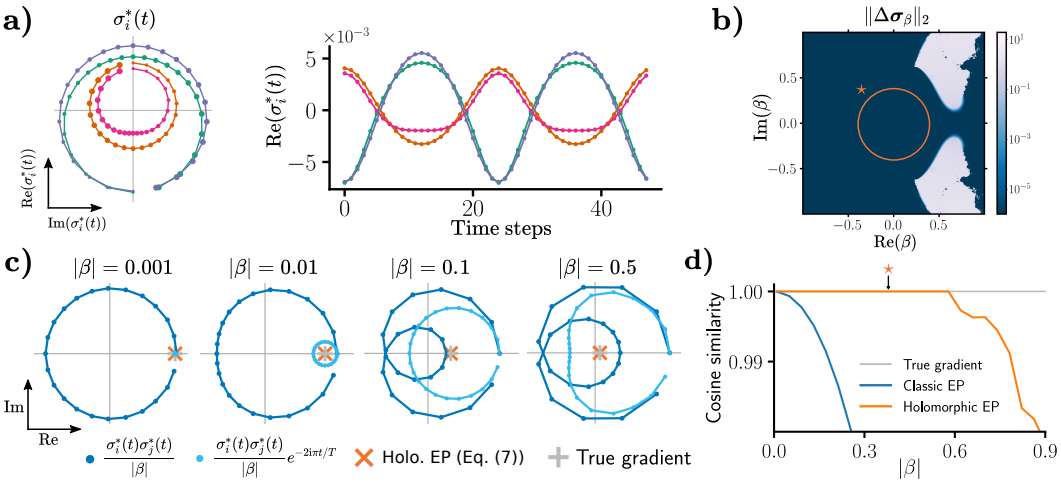

Figure 2: Overview of hEP for a small MLP. **a)** Neural oscillations sampled at $N = 24$ points in the complex plane relative to the free fixed-point in the center (left, point sizes increasing with time). The corresponding real part values over two periods (right). **b)** Map of the euclidean norm between two consecutive steps of the dynamics in the complex plane spanned by β. The map describes the network's dynamical stability. Dark blue corresponds to regions with stable fixed points, while light blue indicates lack of stability. **c)** Adiabatic correlations between two neurons for different $|\beta|$ values (dark blue dots). Filtering the mode over the period T of the teaching oscillation gives the light blue dots, the temporal mean of which (orange cross) is the gradient (grey plus sign). **d)** Cosine similarity with the true gradient of hEP (orange), and classic EP [10] (blue). The $\star$ marks the teaching radius $|\beta|$ of the path in panel b). hEP breaks down when the path passes through unstable (light) regions.

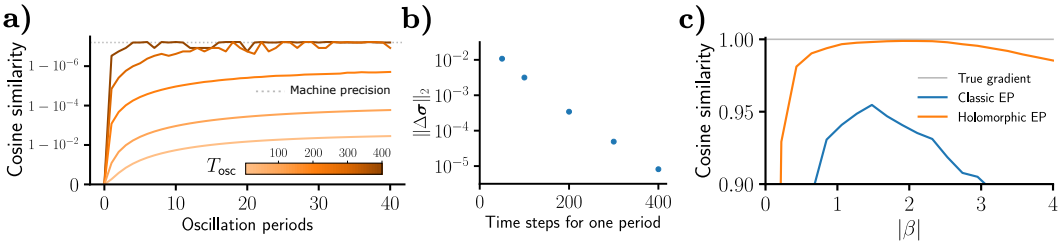

Figure 3: **a)** Cosine similarity between the true gradient obtained with BP through time and the online estimate ($N = 10$) as a function of oscillation periods. Different curves correspond to different oscillation periods (darker color indicates larger T_{osc}). $T_{\mathrm{plas}} \approx 10T_{\mathrm{osc}}$ is enough to accurately estimate the gradient. **b)** Measure of the residual convergence of the network as a function of the oscillation period T_{osc}, showing that $T_{\mathrm{dyn}} \approx 400/10 = 40$ time steps. **c)** Cosine similarity of hEP ($N = 15$) and classic EP with the true gradient as a function of $|\beta|$ with neuronal output noise. All panels used the same MLP with two hidden layers of 256 neurons each, fed with a minibatch of ten MNIST samples.

one teaching oscillation T_{osc}, and the third timescale is the number of periods after which synaptic plasticity occurs T_{plas}. The gradient can be estimated online by:

$$\widetilde{\nabla}(T_{\mathrm{plas}}) := -\frac{1}{T_{\mathrm{plas}}|\beta|}\int_0^{T_{\mathrm{plas}}} \sigma_i(t)\sigma_j(t)e^{-2i\pi t/T_{\mathrm{osc}}}\mathrm{d}t, \tag{10}$$

which converges to the gradient if $T_{\mathrm{dyn}} \ll T_{\mathrm{osc}} \ll T_{\mathrm{plas}}$ (see Appendix A.6). This expression is similar to Eq. (5.2) in [15]. However their teaching signal oscillates discretely between 0 and 1, and therefore produces a biased estimate of the gradient. To test the influence of the oscillation timescale T_{osc} on the online estimate of Eq. (10), we compared the online estimation of the gradient over several periods between several values of T_{osc}. To this end, we used a MLP with two hidden layers with 256 units each, which we fed with a minibatch of MNIST data [31]. We observed that the gradient could be accurately estimated in a few periods for high enough T_{osc} (Fig. 3a, dark curves). However, when the oscillations were too fast, a non-vanishing bias remained in the gradient estimates even for many

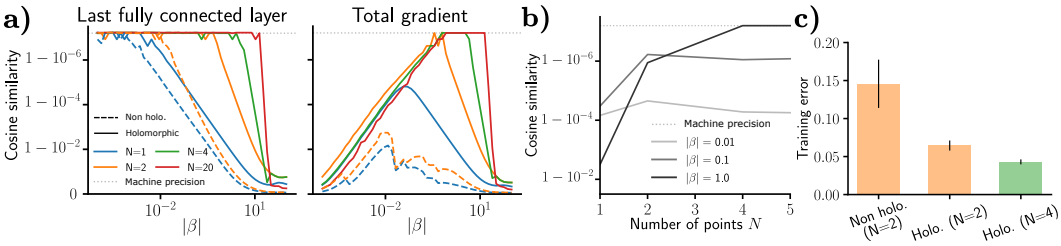

Figure 4: **a)** Cosine similarity between hEP and the true gradient in a holomorphic (plain lines) seven-layer VGG-like CNN [32], and a non-holomorphic version using max pooling and ReLUs (dashed lines). Input data is a minibatch of ImageNet 32×32 [18] consisting of 10 images. The left plot shows a comparison of the gradient with respect to the parameters of network's output layer. The right plot takes into account the gradient with respect to all network parameters. **b)** Cosine similarity in function of N for three teaching amplitude values. Increasing N is only required for higher amplitudes. **c)** Average training error on CIFAR-10. The average is calculated over three random initializations and error bars correspond to two standard deviations.

periods (Fig. 3a, lighter curves). This bias is in all likelihood due to the inability of the system to reach the fixed point (Fig. 3b). Finally, we found that given appropriate period timings, hEP used in the online setting can train a network on MNIST (Table 1). Importantly, the online formulation of hEP allows to dispense with the requirement of strictly separate learning phases by replacing them with separate plasticity mechanisms acting on different timescales.

Table 1: MNIST validation errors in % for classic EP [10], hEP, and online hEP, with and without noise. Results are averages ($n = 3$) $\pm$ stddev. For training errors see Table 4 in Appendix E.

Noise	Class. EP, $\lvert\beta\rvert = 0.1$	Class. EP, $\lvert\beta\rvert = 0.4$	hEP, $\lvert\beta\rvert = 0.4$	Online hEP
Noise-free	1.87 ± 0.01	2.24 ± 0.05	1.97 ± 0.08	2.05 ± 0.02
With noise	88.7 ± 0.0	3.01 ± 0.1	1.96 ± 0.07	1.91 ± 0.16

Finite size teaching oscillations provide robustness to noise. To analyze hEP's robustness to noise, we injected a small-amplitude zero-mean Gaussian noise to each neuron in the network in addition to the input from other neurons. We then used a single minibatch from the MNIST dataset to compute gradient estimates using classic EP and hEP. The latter was computed by using one realisation using $N = 15$ points, whereas the classic EP estimate was computed using the free and nudged fixed points each averaged over $\lceil N/2 \rceil$ to provide a fair comparison. We found that for small β when noise amplitudes were comparable to the activity changes caused by teaching oscillations the gradient estimate diverged from the true gradient of the noise-free system (Fig. 3c). To some extent this effect could be mitigated by choosing a finite teaching signal ($\beta \gg 0$) [12]. However, since β also increases the bias for classic EP this creates a trade-off between choosing β either too small or too large. Importantly, even for the optimal choice of β, classic EP did not accurately approximate the gradient of the noise-free system. In contrast, hEP thanks to its robustness to finite teaching signals did provide an accurate estimate of the gradient despite the noise. We verified that hEP is indeed more robust to noise than classic EP when used to train the network (Table 1). Thus, hEP combined with finite teaching amplitudes provides an effective way for training noisy computational substrates.

Holomorphic EP matches BP performance on large-scale vision benchmarks. To test hEP's ability to train deep neural networks, we first investigated the influence of the number of fixed points N and the teaching amplitude $|\beta|$ on the approximating quality $\hat{\nabla}(N)$ of the loss gradient in a seven-layer VGG-like architecture ([32]; Fig. 4a,b). We ensured holomorphic dynamics by using softmax pooling layers [23] instead of the non-holomorphic max pooling, and by relying on sigmoid weighted linear units (dSiLU) [33] (see Appendix B) instead of standard ReLUs. We first considered a minibatch of ten images from the ImageNet 32×32 dataset [18] and computed the gradient using hEP as well as BPTT for reference. Here, our estimate (Eq. (8)) was computed by first letting the network settle to the free fixed ($\beta = 0$), and then running the phases with complex β. We found that the change to holomorphic dynamics already improved upon the gradient estimates used in previous

work [14, 19]. Moreover, we observed that for the last layer increasing N only extended the range of usable teaching magnitudes $|\beta|$, but did not improve the quality of the gradient estimate. This phenomenon can be understood from Eq. (9), since higher N reduces the bias term considerably, which accommodates higher teaching magnitude $|\beta|$. However, larger amplitudes tended to improve the total gradient, particularly in deep layers where small teaching magnitudes were not enough to produce sufficient error signals (see Appendix C for details). Additionally, larger N were only required when using a higher teaching amplitude (Fig. 4b). Finally, we tested how the gradient quality impacts the network training accuracy on CIFAR-10. We observed that the non-holomorphic VGG was unable to reach low training error (Fig. 4c), which is consistent with the poor gradient quality (Fig 4a). Changing to a holomorphic architecture with the same number of points resulted in a substantial improvement of training accuracy, which was further boosted when training with $N = 4$ consistent with our theory.

Table 2: Validation accuracy of BP and hEP. All values are averages ($n = 3$) $\pm$ stddev.

	CIFAR-10	CIFAR-100		ImageNet 32×32	
	Top-1 (%)	Top-1 (%)	Top-5 (%)	Top-1 (%)	Top-5 (%)
BP	88.3 ± 0.1	62.0 ± 0.5	86.2 ± 0.1	37.2 ± 0.4	60.9 ± 0.1
hEP	88.6 ± 0.2	61.6 ± 0.1	86.0 ± 0.1	36.5 ± 0.3	60.8 ± 0.4

Finally, we wondered whether hEP could train deep neural networks on large-scale datasets, which has remained an open problem for most if not all alternative algorithms to BP [4]. To this end, we trained a five-layer CNN based on the VGG architecture [32] on multiple vision benchmarks including CIFAR-10, CIFAR-100 [17], and the 32×32 pixel version of ImageNet [18], which contains 1.2 million data points and 1000 classes like the full ILSVRC dataset [34]. In all cases we found that the validation accuracy reached by networks trained with BP and hEP using only two fixed points ($N = 2$) were identical within their uncertainties (Table 2). Note that the networks trained with BP were the feed-forward equivalent of the holomorphic networks used with EP, but with ReLUs instead of dSiLU, which did not give satisfactory results. Thus, hEP permits training deep CNNs on ImageNet 32×32 to comparable performance levels as standard BP.

5 Discussion

We have introduced hEP which extends classic EP by computing exact loss gradients through integration over finite size adiabatic neuronal oscillations caused by a teaching signal (Section 3). Importantly, such integration can be accomplished online with purely local learning rules which makes it an exciting theoretical framework for studying learning in the brain where oscillations are ubiquitously observed [35, 36]. In practice we found that numerically evaluating a small number of points during one oscillation cycle provides an excellent gradient approximation that outperforms classic EP and thanks to the finite oscillation amplitude is robust to noise, which is an advantage for training neuromorphic hardware systems [21, 22, 37]. Additionally, the possibility of using finite teaching signals is conducive for training deep CNNs, where infinitesimal teaching signals as used by classic EP, may vanish (Section 4).

A body of previous work has attempted to reconcile BP with neurobiology [3, 38]. EP is most closely related to classic theories of predictive coding (PC) [39–45] which similarly assumes convergent network dynamics cast into an energy minimization problem. PC further assumes that errors are encoded in neuronal dynamics, dedicated dendritic compartments, or separate temporal phases [20, 40, 46, 47]. In a similar vein, Target Propagation (TP) [48–50] assumes locally encoded error signals which in some cases are obtained by iterating approximate inverses, a property reminiscent of EP [51–53] which comes with theoretical guarantees [54]. However, all previous EP studies and most of the works above, with two notable exceptions [38, 50], were all limited to small-scale problems [4]. In contrast, we demonstrate in this article that hEP scales to ImageNet 32×32.

While the ability to run hEP online makes it an appealing model for credit assignment in biological neural networks, this interpretation has several notable shortcomings. First, hEP requires complex-valued neuronal outputs and a holomorphic dynamical system which precludes the use of max pooling and ReLUs and hampers a direct comparison to neurobiology. However, we found that holomorphic

alternatives exists which empirically yield comparable performance. Moreover, complex outputs have a long-standing tradition in computational neuroscience where they appear in variations of Hopfield networks [55–57], in the framework of theta neurons [58], and phasor networks [59] where they are used to describe oscillatory neuronal dynamics. It is possible to interpret hEP within such frameworks. For instance, it is straightforward to interpret complex neuronal output as oscillating activity with a defined amplitude and relative phase to some reference signal accessible to the entire neuronal population. Such a signal could be provided by neuromodulators such as acetylcholine which has been implicated in neural oscillations [60]. Within our framework, the oscillatory teaching signal then corresponds to a slow phase precession between the neuronal activity and the reference. Importantly, such a mechanism implies a hierarchy of oscillation frequencies. Such different oscillations are known to exist in the brain, e.g., theta (4–8 Hz) and gamma (30–70 Hz), but their precise purpose remains elusive. While establishing formal circuit-level equivalences with hEP will require future work, the principled link between oscillatory activity, learning and memory as developed in this article seems promising. Like other algorithms hEP requires symmetric synaptic connectivity between layers which seems biologically implausible. While theoretical guarantees for exact gradient computation are lost without strict symmetry [61], it may not be required for learning [62–64]. Alternatively, symmetry may be acquired through plastic feedback connections [65, 66]. Although our approach neither uses time-varying input nor neuronal spiking dynamics, spiking extensions to EP have been proposed [67, 68]. However, applying our theory to time-varying tasks requiring memory will require additional architectural modifications and theoretical concepts [69] and establishing links to present spike-based approaches [38, 70–72].

Our work augments classic EP with desirable properties for potential neuromorphic applications, which promise power-efficient and equitable artificial intelligence (AI) at the edge and in IoT devices [73–76]. While current software implementation of EP are generally slow compared to backprop, its appeal lies in its potential for training physical networks on future neuromorphic mixed-signal devices that are incompatible with backprop, but achieve settling times on the order of nanoseconds [21, 37]. As exciting as such developments are, they also risk negative societal impacts, e.g., through mass surveillance or allowing AI systems with potentially discriminatory biases to permeate our everyday lives further. A transparent research strategy and taking into account ethical considerations early during product design will be essential to avoid such adverse outcomes. On the upside, our theoretical work further consolidates EP as a conceptual framework for understanding the brain, a fundamental requirement to inform future biomedical research targeted at nervous system disorders.

Acknowledgements

We thank all members of the Zenke Group for comments and discussions. We also thank the anonymous reviewers for their useful feedback. This project was supported by the Swiss National Science Foundation [grant number PCEFP3_202981] and the Novartis Research Foundation.

References

[1] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[3] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[4] 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. *Advances in neural information processing systems*, 31, 2018.

[5] Pritish Narayanan, Alessandro Fumarola, Lucas L Sanches, Kohji Hosokawa, Scott C Lewis, Robert M Shelby, and Geoffrey W Burr. Toward on-chip acceleration of the backpropagation algorithm using nonvolatile memory. *IBM Journal of Research and Development*, 61(4/5):11–1, 2017.

[6] Stefano Ambrogio, Pritish Narayanan, Hsinyu Tsai, Robert M Shelby, Irem Boybat, Carmelo Di Nolfo, Severin Sidler, Massimo Giordano, Martina Bodini, Nathan CP Farinha, et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, 558(7708):60–67, 2018.

[7] Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences*, 111(23):8619–8624, 2014.

[8] Yang Dan and Mu-ming Poo. Spike timing-dependent plasticity of neural circuits. *Neuron*, 44 (1):23–30, 2004.

[9] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural networks*, 111:47–63, 2019.

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

[11] John 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.

[12] Nicolas Zucchet, Simon Schug, Johannes von Oswald, Dominic Zhao, and João Sacramento. A contrastive rule for meta-learning. *arXiv preprint arXiv:2104.01677*, 2021.

[13] Maxence Ernoult, Julie Grollier, Damien Querlioz, Yoshua Bengio, and Benjamin Scellier. Updates of equilibrium prop match gradients of backprop through time in an rnn with static input. *Advances in neural information processing systems*, 32, 2019.

[14] Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

[15] Pierre Baldi and Fernando Pineda. Contrastive learning and neural oscillations. *Neural computation*, 3(4):526–545, 1991.

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

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[18] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[19] Artur Luczak, Bruce L McNaughton, and Yoshimasa Kubo. Neurons learn by predicting future activity. *Nature Machine Intelligence*, pages 1–11, 2022.

[20] James CR Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250, 2019.

[21] Jack Kendall, Ross Pantone, Kalpana Manickavasagam, Yoshua Bengio, and Benjamin Scellier. Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981*, 2020.

[22] Gianluca Zoppo, Francesco Marrone, and Fernando Corinto. Equilibrium propagation for memristor-based recurrent neural networks. *Frontiers in neuroscience*, 14:240, 2020.

[23] A Stergiou, R Poppe, and G Kalliatakis. Refining activation downsampling with softpool. arxiv 2021. *arXiv preprint arXiv:2101.00440*.

[24] Walter Appel. Mathematics for physics and physicists. 2007.

[25] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[26] Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL `http://github.com/deepmind/dm-haiku`.

[27] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[28] Manu Srinath Halvagal and Friedemann Zenke. The combination of hebbian and predictive plasticity learns invariant object representations in deep sensory networks. *bioRxiv*, 2022.

[29] Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.

[30] David Lipshutz, Charles Windolf, Siavash Golkar, and Dmitri Chklovskii. A biologically plausible neural network for slow feature analysis. *Advances in neural information processing systems*, 33:14986–14996, 2020.

[31] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[35] Juergen Fell and Nikolai Axmacher. The role of phase synchronization in memory processes. *Nature reviews neuroscience*, 12(2):105–118, 2011.

[36] Andreas K Engel, Pascal Fries, and Wolf Singer. Dynamic predictions: oscillations and synchrony in top–down processing. *Nature Reviews Neuroscience*, 2(10):704–716, 2001.

[37] Menachem Stern, Daniel Hexner, Jason W Rocks, and Andrea J Liu. Supervised learning in physical networks: From machine learning to learning machines. *Physical Review X*, 11(2): 021045, 2021.

[38] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake A Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature neuroscience*, 24(7):1010–1019, 2021.

[39] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999.

[40] Georg B Keller and Thomas D Mrsic-Flogel. Predictive processing: a canonical cortical computation. *Neuron*, 100(2):424–435, 2018.

[41] Yanping Huang and Rajesh PN Rao. Predictive coding. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(5):580–593, 2011.

[42] 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.

[43] Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv preprint arXiv:2006.04182*, 2020.

[44] Robert Rosenbaum. On the relationship between predictive coding and backpropagation. *Plos one*, 17(3):e0266102, 2022.

[45] Karl Friston. The free-energy principle: a unified brain theory? *Nature reviews neuroscience*, 11(2):127–138, 2010.

[46] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

[47] Paul Haider, Benjamin Ellenberger, Laura Kriener, Jakob Jordan, Walter Senn, and Mihai A Petrovici. Latent equilibrium: Arbitrarily fast computation with arbitrarily slow neurons. *Advances in Neural Information Processing Systems*, 34:17839–17851, 2021.

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

[49] 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.

[50] Maxence Ernoult, Fabrice Normandin, Abhinav Moudgil, Sean Spinney, Eugene Belilovsky, Irina Rish, Blake Richards, and Yoshua Bengio. Towards scaling difference target propagation by learning backprop targets. *arXiv preprint arXiv:2201.13415*, 2022.

[51] Nasir Ahmad, Marcel A van Gerven, and Luca Ambrogioni. Gait-prop: A biologically plausible learning rule derived from backpropagation of error. *Advances in Neural Information Processing Systems*, 33:10913–10923, 2020.

[52] Bill Podlaski and Christian K Machens. Biological credit assignment through dynamic inversion of feedforward networks. *Advances in Neural Information Processing Systems*, 33:10065–10076, 2020.

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

[54] 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.

[55] Akira Hirose. Dynamics of fully complex-valued neural networks. *Electronics letters*, 28(16): 1492–1494, 1992.

[56] Stanislaw Jankowski, Andrzej Lozowski, and Jacek M Zurada. Complex-valued multistate neural associative memory. *IEEE Transactions on neural networks*, 7(6):1491–1496, 1996.

[57] E Paxon Frady and Friedrich T Sommer. Robust computation with rhythmic spike patterns. *Proceedings of the National Academy of Sciences*, 116(36):18050–18059, 2019.

[58] G Bard Ermentrout and Nancy Kopell. Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM journal on applied mathematics*, 46(2):233–253, 1986.

[59] Connor Bybee, E Paxon Frady, and Friedrich T Sommer. Deep learning in spiking phasor neural networks. *arXiv preprint arXiv:2204.00507*, 2022.

[60] M. Hasselmo. The role of acetylcholine in learning and memory. *Curr. Opinion Neurobiol.*, 16: 710–715, 2006.

[61] Benjamin Scellier, Anirudh Goyal, Jonathan Binas, Thomas Mesnard, and Yoshua Bengio. Generalization of equilibrium propagation to vector field dynamics. *arXiv preprint arXiv:1808.04873*, 2018.

[62] 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(1):1–10, 2016.

[63] Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. *Advances in neural information processing systems*, 33:9346–9360, 2020.

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

[65] Mohamed Akrout, Collin Wilson, Peter Humphreys, Timothy Lillicrap, and Douglas B Tweed. Deep learning without weight transport. *Advances in neural information processing systems*, 32, 2019.

[66] Yali Amit. Deep learning with asymmetric connections and hebbian updates. *Frontiers in computational neuroscience*, 13:18, 2019.

[67] Peter O'Connor, Efstratios Gavves, and Max Welling. Training a spiking neural network with equilibrium propagation. In *The 22nd international conference on artificial intelligence and statistics*, pages 1516–1523. PMLR, 2019.

[68] Erwann Martin, Maxence Ernoult, Jérémie Laydevant, Shuai Li, Damien Querlioz, Teodora Petrisor, and Julie Grollier. Eqspike: spike-driven equilibrium propagation for neuromorphic implementations. *Iscience*, 24(3):102222, 2021.

[69] Jack Kendall. A gradient estimator for time-varying electrical networks with non-linear dissipation. *arXiv preprint arXiv:2103.05636*, 2021.

[70] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[71] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):1–15, 2020.

[72] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14:424, 2020.

[73] Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990.

[74] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud, et al. Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5:73, 2011.

[75] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*, 2017.

[76] Julie Grollier, Damien Querlioz, and Mark D Stiles. Spintronic nanodevices for bioinspired computing. *Proceedings of the IEEE*, 104(10):2024–2039, 2016.

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] The theoretical claims refers to Section 3, and the experiments claims to Section 4.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Consent to use ImageNet data was obtained from the official website.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The assumption of Theorem 1 are stated along with the result in Section 3, and in the extended proofs in Appendix A.
   (b) Did you include complete proofs of all theoretical results? [Yes] The proof of Theorem 1 is summarized in Section 3 and a detailed version is in Appendix A.2. The proof of the bias term Eq. (9) is in Appendix A.5. The proof of online estimate is in Appendix A.6
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] The code is included as a link to the public GitHub repository, where instructions to reproduce the results are given (Section 4), and in the supplemental.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The details and hyperparameters of experiments are reported in Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Error bars for training experiments are reported in Table 1 and 2, and in Fig. 4c.
   (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 Section 4 and Appendix F.
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] The creator of the software libraries used for simulations are cited in the beginning of Section 4. The datasets used are cited when mentioned in Section 4.
   (b) Did you mention the license of the assets? [Yes] See Section 4.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (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]

A Theoretical proofs

In this appendix we detail the proofs of the theoretical results in the body text.

A.1 Complex analysis background

We recall here the minimal complex analysis background required to appreciate the theoretical results of this work. In the following, we recall the definitions of holomorphic and Wirtinger derivatives, the Cauchy-Riemann equations and the Cauchy formulas. We refer the reader to Chapter 4 of [S1] for proofs as well as an excellent introduction to complex analysis.

Definition 1 (Holomorphic function)**.** *Let U be an open set of $\mathbb{C}$ and $f : z \in U \mapsto f(z) \in \mathbb{C}$ a function. f is holomorphic at $a \in U$ if the limit*

$$\lim_{z \to a} \frac{f(z) - f(a)}{z - a}$$

exists. This limit is then noted $f'(a)$. f is holomorphic on U if it is holomorphic $\forall a \in U$.

Though this definition looks like the definition of differentiability in $\mathbb{R}$, it brings constraints on the underlying function $\tilde{f} : (x, y) \in \mathbb{R}^2 \mapsto (\mathrm{Re}(f(x + \mathrm{i}y)), \mathrm{Im}(f(x + \mathrm{i}y)))$. The added constraints are the Cauchy-Riemann equations, which can be compactly written after defining Wirtinger derivatives:

Definition 2 (Wirtinger derivatives)**.** *Noting $\partial/\partial x$ and $\partial/\partial y$ the usual partial derivatives in $\mathbb{R}^2$, the Wirtinger derivatives are defined by:*

$$\frac{\partial}{\partial z} := \frac{1}{2}\left(\frac{\partial}{\partial x} - \mathrm{i}\frac{\partial}{\partial y}\right), \qquad \frac{\partial}{\partial \bar{z}} := \frac{1}{2}\left(\frac{\partial}{\partial x} + \mathrm{i}\frac{\partial}{\partial y}\right).$$

In this way, z and its complex conjugate $\bar{z}$ can be thought of as independent variables. We can then state the Cauchy-Riemann equations as:

Theorem 2 (Cauchy-Riemann equations)**.** *if f is holomorphic at $a \in U$, then:*

$$\frac{\partial f}{\partial z}(a) = f'(a), \qquad \frac{\partial f}{\partial \bar{z}}(a) = 0. \tag{11}$$

These constraints ensure that f is locally expandable everywhere in U into a converging power series. In particular, it is differentiable at any order and the derivatives can be computed with the:

Theorem 3 (Cauchy formulas)**.** *Let f be holomorphic on U, let γ be any piece-wise continuously differentiable closed curve in U going around $a \in U$ once and counterclockwise, then:*

$$f^{(n)}(a) = \frac{n!}{2\mathrm{i}\pi} \oint_\gamma \frac{f(z)}{(z-a)^{n+1}} \mathrm{d}z. \tag{12}$$

A.2 Proof of Lemma 1

Here, we give a more detailed proof of holomorphic EP. We recall the:

Lemma 1 (Holomorphic Equilibrium Propagation)**.** *Let F be a scalar function governing the dynamics, so that the holomorphic implicit function theorem can be applied to the fixed point equation $\partial_{\mathbf{s}} F(\boldsymbol{\theta}, \mathbf{s}_0^*, 0) = 0$, then the gradient formula of equilibrium propagation (Eq.* (2)*) holds in the sense of complex differentiation.*

Proof. We first detail precisely the set of equations on which the holomorphic implicit function theorem is applied. At the free fixed point $(\boldsymbol{\theta} = \boldsymbol{\theta}_0, \beta = 0)$ that which exists by assumption, we have the following set of equations:

$$\frac{\partial F}{\partial s_j}(\boldsymbol{\theta}_0, \mathbf{s}_0^*, 0) = 0, \quad 1 \leq j \leq n,$$

where n is the number of units in the system. The functions $\partial_{s_j} F$ are holomorphic by assumption. If we further assume that the Hessian of F with respect to $\mathbf{s}$ is invertible in $(\boldsymbol{\theta}_0, \mathbf{s}_0^*, 0)$, i.e.:

$$\det\left(\frac{\partial^2 F}{\partial s_i \partial s_j}(\boldsymbol{\theta}_0, \mathbf{s}_0^*, 0)\right)_{i,j} \neq 0,$$

then the holomorphic version of the implicit function theorem [S2] can be applied and there exists an open neighbourhood of $(\boldsymbol{\theta} = \boldsymbol{\theta}_0, \beta = 0)$ in the complex domain where the implicit map $(\boldsymbol{\theta}, \beta) \mapsto \mathbf{s}^*_{\boldsymbol{\theta},\beta}$ is holomorphic, and where the fixed point equations hold:

$$\frac{\partial F}{\partial \mathbf{s}}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta) = 0.$$

At such fixed points, we have that the total derivatives of F with respect to either β or $\boldsymbol{\theta}$ are equal to the partial derivatives, which can be seen by applying the chain rule of complex differentiation using Wirtinger derivatives. There are now in principle three contributions to the total derivative of F with respect to β:

$$\frac{\mathrm{d}F}{\mathrm{d}\beta}(\boldsymbol{\theta}, \mathbf{s}_{\boldsymbol{\theta},\beta}, \beta) = \frac{\partial F}{\partial \beta}(\boldsymbol{\theta}, \mathbf{s}_{\boldsymbol{\theta},\beta}, \beta) + \underbrace{\frac{\partial F}{\partial \mathbf{s}}}_{=0 \text{ at a fixed point}} \cdot \frac{\partial \mathbf{s}}{\partial \beta}(\boldsymbol{\theta}, \beta) + \underbrace{\frac{\partial F}{\partial \overline{\mathbf{s}}}}_{=0 \text{ by Cauchy-Riemann (Eq. (11))}} \cdot \frac{\partial \overline{\mathbf{s}}}{\partial \beta}(\boldsymbol{\theta}, \beta), \tag{13}$$

where $\overline{\mathbf{s}}$ denotes the complex conjugate of $\mathbf{s}$. At the fixed point however, the second term on the right hand side cancels by definition. The third term is zero because F is holomorphic, i.e., its derivative with respect to the conjugate variable $\overline{\mathbf{s}}$ is zero according to the Cauchy-Riemann condition [S1]. The same argument holds for the total derivative with respect to $\boldsymbol{\theta}$.

Finally, the cross-derivatives of F with respect to complex β and $\boldsymbol{\theta}$ can be exchanged, which is a consequence of the Schwarz theorem applied to the function $(\boldsymbol{\theta}, \beta) \mapsto G(\boldsymbol{\theta}, \beta) := F(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta)$. Therefore we have that:

$$\begin{aligned}
\frac{\partial^2 G}{\partial \beta \partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \beta) &= \frac{\partial^2 G}{\partial \boldsymbol{\theta} \partial \beta}(\boldsymbol{\theta}, \beta), \\
\frac{\mathrm{d}}{\mathrm{d}\beta}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} F(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta) &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\frac{\mathrm{d}}{\mathrm{d}\beta} F(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta), \\
\frac{\mathrm{d}}{\mathrm{d}\beta}\frac{\partial}{\partial \boldsymbol{\theta}} F(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta) &= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\frac{\partial}{\partial \beta} F(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta), \quad \text{by Eq. (13).}
\end{aligned}$$

By then applying this equality in $\beta = 0$ and $\boldsymbol{\theta} = \boldsymbol{\theta}_0$, we obtain the EP gradient formula (Eq. (2)) for complex differentiation:

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\bigg|_{\beta=0}\left(\frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta)\right) = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\frac{\partial F}{\partial \beta}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta) = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}},$$

which concludes the proof. □

A.3 Proof of Theorem 1

Theorem 1 (Exact gradient from finite teaching signals)**.** *Assuming that the conditions of Lemma 1 are met and let $|\beta| > 0$ be the radius of a circular path around 0 in $\mathbb{C}$ contained in the open set U on which the fixed point $\mathbf{s}^*_{\boldsymbol{\theta},\beta}$ is defined. Further assume that this path is parameterized by $t \in [0, T] \mapsto \beta(t) = |\beta| e^{2\mathrm{i}\pi t/T}$, where i is the imaginary unit. Then the loss gradient is given by:*

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \frac{1}{T|\beta|}\int_0^T \frac{\partial F}{\partial \boldsymbol{\theta}}\left(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta(t)}, \beta(t)\right) e^{-2\mathrm{i}\pi t/T}\mathrm{d}t \quad . \tag{14}$$

Proof. By assumption the fixed point $\beta \mapsto \mathbf{s}^*_{\boldsymbol{\theta},\beta}$ is defined on an open set U (by the holomorphic implicit function theorem) containing the disk of radius $|\beta|$ centered around 0. In particular, the function $\beta \in U \mapsto \partial_{\boldsymbol{\theta}} F(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta)$, is also holomorphic by composition. The left hand side of Eq. (4) can thus be computed with the Cauchy formulas (Eq. (12) with $f = \partial_{\boldsymbol{\theta}} F$, $n = 1$, $a = 0$), and γ an arbitrary closed path leading around zero once and counterclockwise in U:

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\bigg|_{\beta=0}\left(\frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta)\right) = \frac{1}{2\mathrm{i}\pi}\oint_\gamma \frac{1}{\beta^2}\frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta)\mathrm{d}\beta \quad . \tag{15}$$

To obtain Eq. (14), we choose γ as a circular path in the complex plane with radius $|\beta| > 0$ parameterized by time $t \in [0, T] \mapsto \beta(t) = |\beta| e^{2\mathrm{i}\pi t/T}$, where T is a full period. After the change of variable $\mathrm{d}\beta = \left(2\mathrm{i}\pi\beta(t)/T\right)\mathrm{d}t$ in Eq. (15), and using Lemma 1, the loss gradient is given by:

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} &= \frac{1}{2\mathrm{i}\pi} \oint_\gamma \frac{1}{\beta^2} \frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta}, \beta)\mathrm{d}\beta \\
&= \frac{1}{2\mathrm{i}\pi} \int_0^T \frac{1}{\beta(t)^2} \frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta(t)}, \beta(t)) \left(\frac{2\mathrm{i}\pi\beta(t)}{T}\right) \mathrm{d}t \\
&= \frac{1}{T} \int_0^T \frac{1}{\beta(t)} \frac{\partial F}{\partial \boldsymbol{\theta}} \left(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta(t)}, \beta(t)\right) \mathrm{d}t \\
&= \frac{1}{T|\beta|} \int_0^T \frac{\partial F}{\partial \boldsymbol{\theta}} \left(\boldsymbol{\theta}, \mathbf{s}^*_{\boldsymbol{\theta},\beta(t)}, \beta(t)\right) e^{-2\mathrm{i}\pi t/T} \mathrm{d}t.
\end{aligned}$$

□

A.4 Roles of real and imaginary parts in the learning rule

Recall that for the continuous Hopfield network case the partial derivative of F with respect to a parameter w_{ij} is the product of pre and post activation (Eq. (6)), so that applying Eq. (4) yields:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} = \frac{\mathrm{d}}{\mathrm{d}\beta}\bigg|_{\beta=0} \underbrace{\left(\frac{\partial F}{\partial w_{ij}}(\boldsymbol{\theta}, \mathbf{s}^*_\beta, \beta)\right)}_{=-\sigma(s^*_{i,\beta})\sigma(s^*_{j,\beta})} = -\left.\frac{\mathrm{d}\left(\sigma(s^*_{i,\beta})\sigma(s^*_{j,\beta})\right)}{\mathrm{d}\beta}\right|_{\beta=0},$$

which can further be expressed as:

$$\left.\frac{\mathrm{d}\left(\sigma(s^*_{i,\beta})\sigma(s^*_{j,\beta})\right)}{\mathrm{d}\beta}\right|_{\beta=0} = \left.\left(\sigma(s^*_{i,\beta})\frac{\mathrm{d}\sigma(s^*_{j,\beta})}{\mathrm{d}\beta}\right)\right|_{\beta=0} + \left.\left(\sigma(s^*_{j,\beta})\frac{\mathrm{d}\sigma(s^*_{i,\beta})}{\mathrm{d}\beta}\right)\right|_{\beta=0}. \tag{16}$$

Using the same assumptions as Section 3, the map $\beta \in U \mapsto s^*_{i,\beta}$ is holomorphic, and so is the map $\beta \in U \mapsto \sigma(s^*_{i,\beta})$ by composition. We can thus expand it in a power series around zero:

$$\sigma(s^*_{i,\beta}) = \sum_{k=0}^{\infty} \frac{\beta^k}{k!} \left.\frac{\mathrm{d}^k\sigma(s^*_{i,\beta})}{\mathrm{d}\beta^k}\right|_{\beta=0}.$$

We can then separate the sum into the real and imaginary parts because the series converge absolutely. Assuming that $\beta = |\beta|e^{2\mathrm{i}\pi t/T}$, and applying the Euler formula, we obtain:

$$\begin{aligned}
\mathrm{Re}\left(\sigma(s^*_{i,\beta})\right) &= \sum_{k=0}^{\infty} \cos\left(\frac{2k\pi t}{T}\right) \frac{|\beta|^k}{k!} \left.\frac{\mathrm{d}^k\sigma(s^*_{i,\beta})}{\mathrm{d}\beta^k}\right|_{\beta=0}, \\
\mathrm{Im}\left(\sigma(s^*_{i,\beta})\right) &= \sum_{k=1}^{\infty} \sin\left(\frac{2k\pi t}{T}\right) \frac{|\beta|^k}{k!} \left.\frac{\mathrm{d}^k\sigma(s^*_{i,\beta})}{\mathrm{d}\beta^k}\right|_{\beta=0}.
\end{aligned} \tag{17}$$

Therefore, the first derivative ($k = 1$) with respect to β in Eq. (16) can be obtained by either projecting the real part against the cosine function, or imaginary part against the sine function:

$$\begin{aligned}
\left.\frac{\mathrm{d}\sigma(s^*_{i,\beta})}{\mathrm{d}\beta}\right|_{\beta=0} &= \frac{2}{|\beta|T} \int_0^T \mathrm{Re}\left(\sigma(s^*_{i,\beta})\right) \cos\left(\frac{2\pi t}{T}\right)\mathrm{d}t, \\
&= \frac{2}{|\beta|T} \int_0^T \mathrm{Im}\left(\sigma(s^*_{i,\beta})\right) \sin\left(\frac{2\pi t}{T}\right)\mathrm{d}t,
\end{aligned}$$

by orthogonality of the family $\left((t \mapsto \cos\left(\frac{2k\pi t}{T}\right))_{k\geq 0}, (t \mapsto \sin\left(\frac{2k\pi t}{T}\right))_{k\geq 0}\right)$ in $L^2[0, T]$. Note that the higher order derivatives with respect to β can be obtained as well by projecting against the corresponding cosine or sine function. The same holds for index j by symmetry. As an interesting final note, if we define $\mathrm{Re}_1(\sigma(s^*_{i,\beta}))$ and $\mathrm{Im}_1(\sigma(s^*_{i,\beta}))$, the first order contributions in β to the real and imaginary parts of the neural activity, we find that they are the only ones to contribute to the gradient computation. We can appreciate that they are related through $\mathrm{Im}_1(\sigma(s^*_{i,\beta})) = -\frac{T}{2\pi}\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{Re}_1(\sigma(s^*_{i,\beta}))$, where the time derivative is at the scale of the teaching signal.

A.5 Derivation of the bias term

Recall the definition of $\beta_k := |\beta| e^{2\mathrm{i}\pi k/N}$, for $k \in [0, ..., N-1]$, $N \geq 2$, and the gradient estimate (Eq. (8)):

$$\hat{\nabla}(N) := \frac{1}{N|\beta|} \sum_{k=0}^{N-1} \frac{\partial F}{\partial \boldsymbol{\theta}} \left(\boldsymbol{\theta}, \mathbf{s}^*_{\beta_k}, \beta_k\right) e^{-2\mathrm{i}\pi k/N}.$$

For simplicity of notation, we rewrite $\partial_{\boldsymbol{\theta}} F(\beta) := \frac{\partial F}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}, \mathbf{s}^*_\beta, \beta)$. The function $\beta \mapsto \partial_{\boldsymbol{\theta}} F(\beta)$ is holomorphic on an open set U including zero, and so is $\beta \mapsto \mathbf{s}^*_\beta$ by the holomorphic implicit function theorem. We assume the β_k are included in U, so that we can expand $\partial_{\boldsymbol{\theta}} F(\beta_k)$ in a power series around zero:

$$\partial_{\boldsymbol{\theta}} F(\beta_k) = \sum_{p=0}^{\infty} \frac{\beta_k^p}{p!} \left[\frac{\mathrm{d}^p}{\mathrm{d}\beta^p} \partial_{\boldsymbol{\theta}} F\right](0),$$

we define $C_p := \left[\frac{\mathrm{d}^p}{\mathrm{d}\beta^p} \partial_{\boldsymbol{\theta}} F\right](0)$. The quantity of interest is C_1, since it is the gradient of the loss (Eq. (4))

$$\begin{aligned}
\partial_{\boldsymbol{\theta}} F(\beta_k) &= C_0 + \beta_k C_1 + \sum_{p=2}^{\infty} \frac{\beta_k^p}{p!} C_p \\
\frac{\partial_{\boldsymbol{\theta}} F(\beta_k)}{\beta_k} &= C_0 \beta_k^{-1} + C_1 + \sum_{p=2}^{\infty} \frac{\beta_k^{p-1}}{p!} C_p \\
\frac{1}{N} \sum_{k=0}^{N-1} \frac{\partial_{\boldsymbol{\theta}} F(\beta_k)}{\beta_k} &= C_1 + C_0 \frac{1}{N} \sum_{k=0}^{N-1} \beta_k^{-1} + \frac{1}{N} \sum_{k=0}^{N-1} \sum_{p=2}^{\infty} \frac{\beta_k^{p-1}}{p!} C_p.
\end{aligned}$$

The sum symbols on the right can be interchanged thanks to the absolute convergence of the power series.

$$\begin{aligned}
\frac{1}{N} \sum_{k=0}^{N-1} \frac{\partial_{\boldsymbol{\theta}} F(\beta_k)}{\beta_k} &= C_1 + C_0 \frac{1}{N} \sum_{k=0}^{N-1} \beta_k^{-1} + \sum_{p=2}^{\infty} \frac{C_p}{p!} \frac{1}{N} \sum_{k=0}^{N-1} \beta_k^{p-1} \\
\frac{1}{N} \sum_{k=0}^{N-1} \frac{\partial_{\boldsymbol{\theta}} F(\beta_k)}{\beta_k} &= C_1 + C_0 \frac{1}{N|\beta|} \sum_{k=0}^{N-1} e^{-2\mathrm{i}\pi k/N} + \sum_{p=1}^{\infty} \frac{C_{p+1}}{(p+1)!} \frac{|\beta|^p}{N} \sum_{k=0}^{N-1} e^{2\mathrm{i}\pi p k/N}.
\end{aligned}$$

It remains to evaluate the geometric sums of the form $\sum_{k=0}^{N-1} e^{2\mathrm{i}\pi p k/N}$ for $p = -1$ and $p \geq 1$. If N divides p, i.e $p \equiv 0\ (N)$, then we can write $p = Nq$ and we have:

$$\sum_{k=0}^{N-1} e^{2\mathrm{i}\pi p k/N} = \sum_{k=0}^{N-1} e^{2\mathrm{i}\pi q N k/N} = \sum_{k=0}^{N-1} e^{2\mathrm{i}\pi q k} = \sum_{k=0}^{N-1} 1 = N.$$

If N does not divide p, then the geometric sum of ratio $e^{2\mathrm{i}\pi p/N}$ can be computed:

$$\sum_{k=0}^{N-1} e^{2\mathrm{i}\pi p k/N} = \frac{1 - (e^{2\mathrm{i}\pi p/N})^N}{1 - e^{2\mathrm{i}\pi p/N}} = \frac{1 - e^{2\mathrm{i}\pi p}}{1 - e^{2\mathrm{i}\pi p/N}} = \frac{1 - 1}{1 - e^{2\mathrm{i}\pi p/N}} = 0.$$

We thus have that:

$$\begin{aligned}
\frac{1}{N} \sum_{k=0}^{N-1} \frac{\partial_{\boldsymbol{\theta}} F(\beta_k)}{\beta_k} &= C_1 + C_0 \frac{1}{N|\beta|} \underbrace{\sum_{k=0}^{N-1} e^{-2\mathrm{i}\pi k/N}}_{=0} + \sum_{p=1}^{\infty} \frac{C_{p+1}}{(p+1)!} \frac{|\beta|^p}{N} \underbrace{\sum_{k=0}^{N-1} e^{2\mathrm{i}\pi p k/N}}_{=0 \text{ when } p \not\equiv 0\ (N)} \\
\frac{1}{N} \sum_{k=0}^{N-1} \frac{\partial_{\boldsymbol{\theta}} F(\beta_k)}{\beta_k} &= C_1 + \sum_{p \equiv 0\ (N)}^{\infty} \frac{C_{p+1} |\beta|^p}{(p+1)!},
\end{aligned}$$

which is the result of Eq. (9).

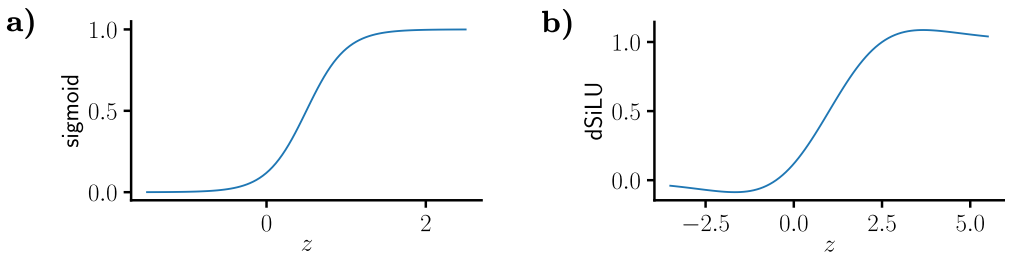

Figure 5: **a)** The shifted sigmoid we used in multi-layered perceptrons experiments. **b)** The dSiLU we used in CNNs experiments.

A.6 Derivation of the online estimate

Recall the formula of the online estimate (Eq. (10)):

$$\widetilde{\nabla}(T_{\text{plas}}) := -\frac{1}{T_{\text{plas}}|\beta|}\int_0^{T_{\text{plas}}} \sigma_i(t)\sigma_j(t)e^{-2\mathrm{i}\pi t/T_{\text{osc}}}\mathrm{d}t.$$

If $T_{\text{dyn}} \ll T_{\text{osc}}$, the product of activities can be replaced by its value at the fixed point, and an exact gradient is computed after each period (Eq. (7)). Then if $T_{\text{osc}} \ll T_{\text{plas}}$, the integral can be divided into an integer amount of completed periods plus a remainder: $T_{\text{plas}} = kT_{\text{osc}} + T_{\text{rem}}$, where $k \in \mathbb{N}$ and $T_{\text{rem}} < T_{\text{osc}}$. We then have by periodicity that:

$$\widetilde{\nabla}(T_{\text{plas}}) = \underbrace{\frac{kT_{\text{osc}}}{kT_{\text{osc}} + T_{\text{rem}}}}_{\to 1 \text{ when } T_{\text{plas}}\to\infty} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w_{ij}} - \underbrace{\frac{1}{kT_{\text{osc}} + T_{\text{rem}}}}_{\to 0 \text{ when } T_{\text{plas}}\to\infty} \frac{1}{|\beta|}\int_0^{T_{\text{rem}}} \sigma_i^*(t)\sigma_j^*(t)e^{-2\mathrm{i}\pi t/T_{\text{osc}}}\mathrm{d}t.$$

In this way, when averaging over large T_{plas}, the number of completed cycles outweighs the current period. Thus, by simply averaging over many oscillation cycles allows estimating gradients without explicit separate phases.

B Detailed architecture

B.1 Dynamics for multi-layer perceptrons

Assuming a number of L layers, we note $\mathbf{s}_l$ the subset of units in layer l, with $\mathbf{s}_0 = \mathbf{x}$ and $\mathbf{y}$ the one hot class label. We note $\mathbf{W}_l$, and $\mathbf{b}_l$ the weight and biases of layer $l \geq 1$. The energy function F for a MLP optimizing the cross entropy loss reads:

$$F(\boldsymbol{\theta}, \mathbf{s}, \beta, \mathbf{y}) = \sum_{l=1}^{L-1}\left[\frac{1}{2}\|\mathbf{s}_l\|^2 - \sigma(\mathbf{s}_{l-1})^\top \cdot \mathbf{W}_l \cdot \sigma(\mathbf{s}_l) - \mathbf{b}_l^\top \cdot \sigma(\mathbf{s}_l)\right] - \beta\mathbf{y}^\top \cdot \log(\mathbf{s}_L).$$

The activation function used for multi-layer perceptrons is the shifted sigmoid $z \mapsto 1/(1+e^{-4z+2})$ (Fig. 5a). We use the layer-wise discrete dynamics introduced by [S3, S4], which read:

$$\begin{cases} \mathbf{s}_l & \leftarrow \sigma\left(\mathbf{W}_l\mathbf{s}_{l-1} + \mathbf{W}_{l+1}{}^\top\mathbf{s}_{l+1} + \mathbf{b}_l + \boldsymbol{\eta}_l\right), & \text{for } 1 \leq l \leq L-2 \\ \mathbf{s}_{L-1} & \leftarrow \sigma\left(\mathbf{W}_{L-1}\mathbf{s}_{L-2} + \beta\mathbf{W}_L{}^\top(\mathbf{y} - \mathbf{s}_L) + \mathbf{b}_{L-1} + \boldsymbol{\eta}_{L-1}\right), \\ \mathbf{s}_L & \leftarrow \mathrm{Softmax}\left(\mathbf{W}_L\mathbf{s}_{L-1} + \mathbf{b}_L\right), \end{cases}$$

where $\boldsymbol{\eta}_l$ is an optional Gaussian noise added for Fig. 3c and Table 1. The noise was sampled at each time step.

B.2 Dynamics for convolutional neural networks

The activation function used for CNNs is a sigmoid-weighted linear unit [S5] (Fig. 5b):

$$\mathrm{dSiLU}(z) := \left(\frac{z}{2}\right)\frac{1}{1+e^{-z}} + \left(1 - \frac{z}{2}\right)\frac{1}{1+e^{-z+2}}.$$

We denote by $\mathcal{P}$ the pooling operation, and $\tilde{\mathcal{P}}$ the corresponding unpooling operation. '$*$' denotes the convolution when preceded by $\mathbf{W}$ and transpose convolution when preceded by $\mathbf{W}^\top$. The energy function F for a CNN optimizing the cross entropy loss reads:

$$F(\boldsymbol{\theta}, \mathbf{s}, \beta, \mathbf{y}) = \sum_{l \in \{\text{Conv}\}} \left[\frac{1}{2} \|\mathbf{s}_l\|^2 - \sigma(\mathbf{s}_{l-1})^\top \cdot \mathcal{P}(\mathbf{W}_l * \sigma(\mathbf{s}_l)) - \mathbf{b}_l^\top \cdot \sigma(\mathbf{s}_l) \right]$$
$$\sum_{l \in \{\text{FC}\}} \left[\frac{1}{2} \|\mathbf{s}_l\|^2 - \sigma(\mathbf{s}_{l-1})^\top \cdot \mathbf{W}_l \cdot \sigma(\mathbf{s}_l) - \mathbf{b}_l^\top \cdot \sigma(\mathbf{s}_l) \right] - \beta \mathbf{y}^\top \cdot \log(\mathbf{s}_L).$$

We use the layer-wise discrete dynamics introduced by [S3, S4], which read:

$$\begin{cases} \mathbf{s}_l & \leftarrow \sigma\left(\mathcal{P}(\mathbf{W}_l * \mathbf{s}_{l-1}) + \mathbf{W}_{l+1}^\top * \tilde{\mathcal{P}}(\mathbf{s}_{l+1}) + \mathbf{b}_l\right), & \text{for } l \in \{\text{Conv layers}\} \\ \mathbf{s}_l & \leftarrow \sigma\left(\mathbf{W}_l \mathbf{s}_{l-1} + \mathbf{W}_{l+1}{}^\top \mathbf{s}_{l+1} + \mathbf{b}_l\right), & \text{for } l \in \{\text{FC layers}\} \\ \mathbf{s}_{L-1} & \leftarrow \sigma\left(\mathbf{W}_{L-1} \mathbf{s}_{L-2} + \beta \mathbf{W}_L{}^\top (\mathbf{y} - \mathbf{s}_L) + \mathbf{b}_{L-1}\right), & \\ \mathbf{s}_L & \leftarrow \text{Softmax}(\mathbf{W}_L \mathbf{s}_{L-1} + \mathbf{b}_L). & \end{cases}$$

We used Softmax pooling [S6] with a tunable temperature τ, instead of the non-holomorphic Max pooling. The output y of Softmax pooling of an input $\mathbf{x}$ is defined for a kernel neighbourhood $\mathbf{R}$ by:

$$y = \sum_{i \in \mathbf{R}} \left(\frac{e^{x_i/\tau}}{\sum_{j \in \mathbf{R}} e^{x_j/\tau}} \right) x_i.$$

Note that Softmax pooling interpolates between Average pooling ($\tau \to \infty$) and Max pooling ($\tau \to 0$).

C Layer-wise comparison of the gradient in a deep network

Here we show in Fig. 6 the complete layer-wise cosine similarities between the estimates of holomorphic EP for various N and the true gradient computed by automatic differentiation.

D Dynamical stability in the complex plane

We show in Fig. 7 how the area in $\mathbb{C}$ where the fixed point empirically exists varies with different architecture choices. The data used for each panel is a digit from the MNIST dataset. As in Fig.2b), dark blue means that the fixed point exists, whereas light areas denote divergence. These diverging areas could be due to the poles of the activation functions used. For example, the sigmoid function $z \mapsto 1/(1 + e^{-z})$ has $\{(2k+1)\mathrm{i}\pi; k \in \mathbb{Z}\}$ as a set of poles where it diverges. Although we did not systematically study this phenomenon in this work, we strongly suspect that these unstable areas are partly the result of the teaching signal being too strong or the weights being poorly distributed, thereby driving the complex neural activities near to the poles. To some extent, the poles can be brought farther by introducing a coefficient in the exponential, but it results in flatter activation functions on the real axis, so a trade-off should be found. In practice, we found that choosing reasonably the activation function, weight initialization, and teaching radius $|\beta|$ lead to enough stable areas around 0 to compute the gradient.

E Hyperparameters

E.1 MNIST experiments

The digits were rescaled by 255 and flattened. The hyperparameters used for training are reported in Table 3, and the training errors are reported in Table 4.

E.2 Large-scale experiments

In the training experiments for CIFAR-10, CIFAR-100, and ImageNet 32×32, the training data was normalized, then augmented with 50% chance random horizontal flips, resized to 36×36 resolution

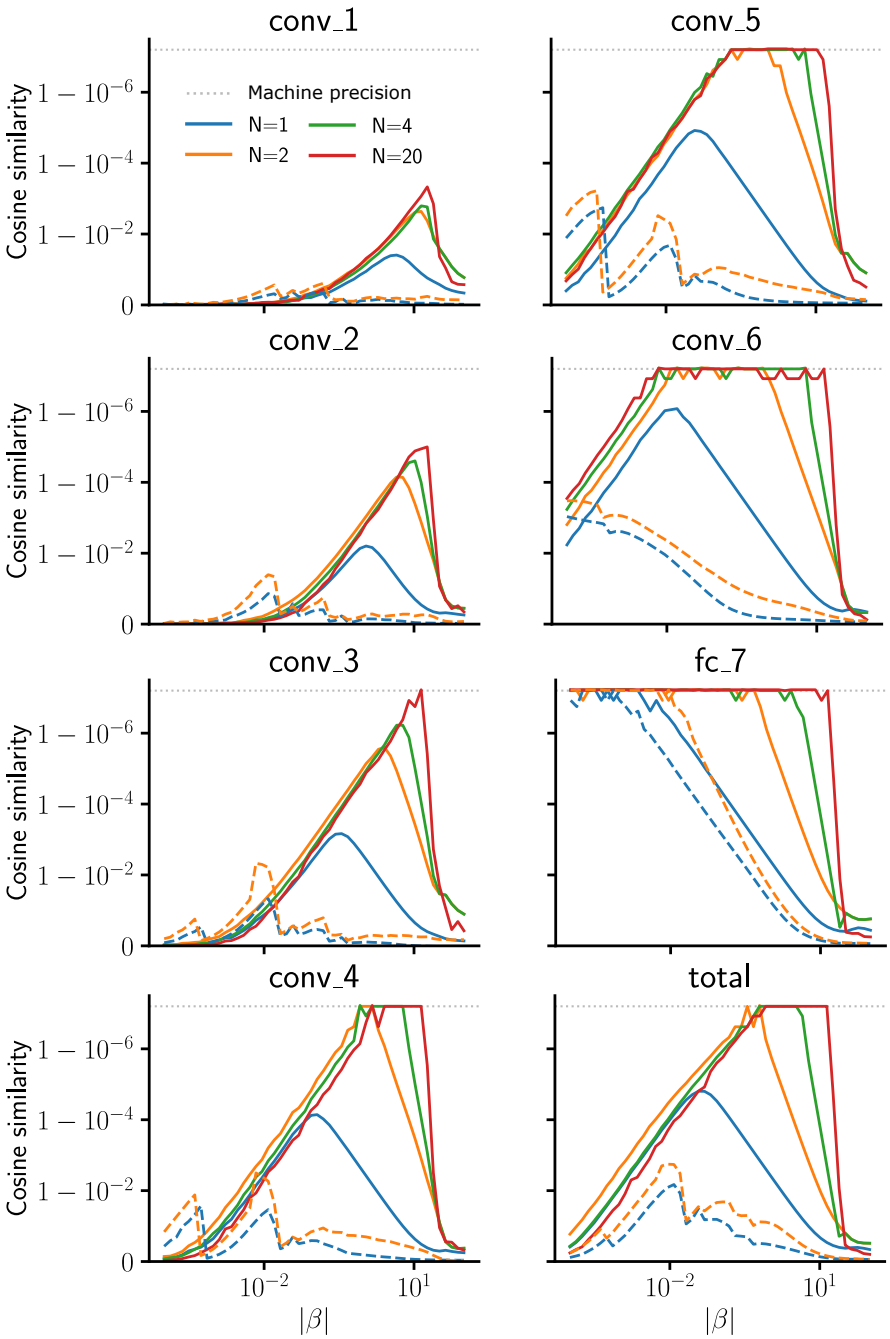

Figure 6: The complete layer-wise cosine similarity of Fig. 4a).

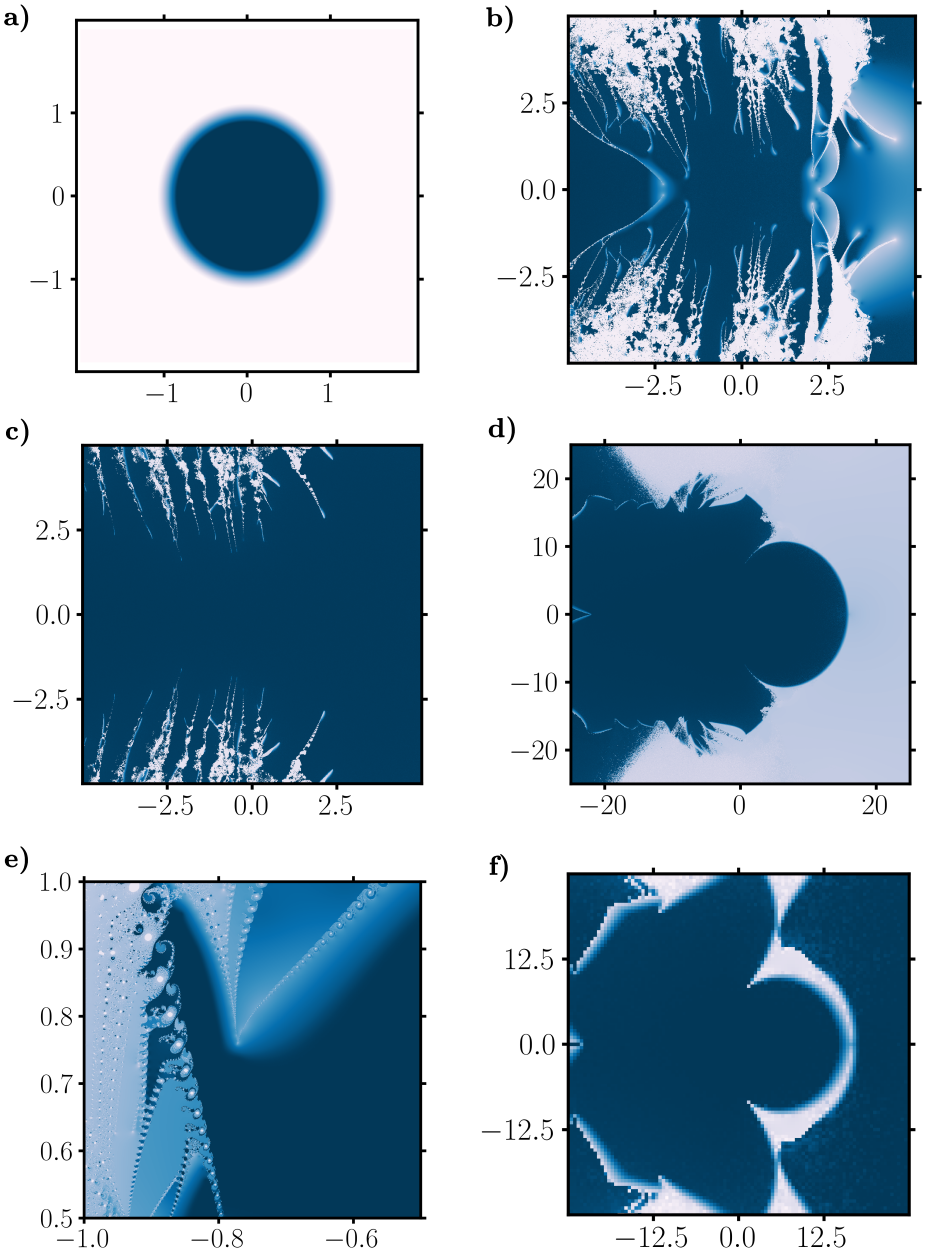

Figure 7: Map of convergence to a fixed point for complex β in various settings. **a)** MLP with linear activation function and low weight initialization. **b)** MLP with shifted sigmoid activation and default weight initialization. **c)** Same as b) but with reduced weight initialization. **d)** Same as b) but with dSiLU activation function. **e)** Zoom at a frontier between stable and unstable regions. **f)** Small CNN with dSiLU activation and Softmax pooling.

Table 3: Hyperparameters used for the MNIST training experiment of Table 1.

Hyperparameter	Classic EP	hEP	Online hEP
Batch size	20	20	20
Learning rate	5e-2	5e-2	5e-2
Epochs	50	50	50
$\lvert\beta\rvert$	0.1 and 0.4	0.4	0.4
T_{free}	350	200	200*
T_{nudge}	350	50	N/A
T_{osc}	N/A	N/A	300
T_{plas}	N/A	N/A	900
N	N/A	10	10
Noise**	4e-2	4e-2	4e-2

* Only used for evaluation

** Standard deviation of the Gaussian noise for experiments with noise.

Table 4: MNIST training and validation errors for classic EP [10], hEP, and online hEP, with and without noise. All results are averages ($n = 3$) $\pm$ one standard deviation.

	Class. EP, $\lvert\beta\rvert = 0.1$		Class. EP, $\lvert\beta\rvert = 0.4$		hEP, $\lvert\beta\rvert = 0.4$		Online hEP	
Noise	Train (%)	Val (%)	Train (%)	Val (%)	Train (%)	Val (%)	Train (%)	Val. (%)
No	$0.05^{\pm 0.02}$	$1.87^{\pm 0.01}$	$0.19^{\pm 0.05}$	$2.24^{\pm 0.05}$	$0.02^{\pm 0.01}$	$1.97^{\pm 0.08}$	$0.11^{\pm 0.01}$	$2.05^{\pm 0.02}$
Yes	$88.8^{\pm 0.0}$	$88.7^{\pm 0.0}$	$1.96^{\pm 0.2}$	$3.01^{\pm 0.1}$	$0.14^{\pm 0.03}$	$1.96^{\pm 0.07}$	$0.13^{\pm 0.03}$	$1.91^{\pm 0.16}$

with padding, and cropped randomly back to 32×32. The optimizer used was stochastic gradient descent with momentum. Pooling was applied at all layers for the five-layer CNN, and every other layer starting with the first layer in the seven-layer CNN.

Table 5: Hyperparameters used for the VGG training experiments of Table 2 and Fig.4c.

Hyperparameter	CIFAR-10	CIFAR-100	ImageNet 32×32	CIFAR-10 (Fig.4c)
Batch size	128	128	256	128
Channel sizes	[128, 256, 512, 512]			[128, 128, 256, 256, 512, 512]
Kernel sizes	[3, 3, 3, 3]			[3, 3, 3, 3, 3, 3]
Strides	[1, 1, 1, 1]			[1, 1, 1, 1, 1, 1]
Paddings	[1, 1, 1, 0]			[1, 1, 1, 0, 1, 0]
SoftPool window	2×2			2×2
SoftPool stride	2			2
SoftPool temp.	1			10
Initial LRs*	[25, 15, 10, 8, 5] $\times$ 1e-2			[5, 4, 4, 3, 3, 2, 2] $\times$ 1e-2
Final LRs	[25, 15, 10, 8, 5] $\times$ 1e-9			[5, 4, 4, 3, 3, 2, 2] $\times$ 1e-9
Weight decay	2e-3	1e-2	5e-4	[5, 5, 5, 5, 5, 5, 10] $\times$ 1e-4
Momentum	0.9	0.9	0.9	0.9
Epochs	90	90	90	90
$\lvert\beta\rvert$	1.0	1.0	1.0	1.0
T_{free}	250	250	250	260
T_{nudge}	60	60	60	60
N	2	2	2	2 and 4

* Learning rates were decayed with cosine annealing without restart [S7].

F Simulations details

All simulations were performed on an in-house GPU cluster or workstations. Each simulation in Table 2 was run in parallel on four NVIDIA V100 GPUs. The training runs on ImageNet 32×32 took 5.5 days each for EP, and a few hours for BP. The runs on CIFAR-10 and CIFAR-100 took one

day on average depending on the architecture (5 or 7 layers) and the number of time steps used for the dynamics. The use of complex numbers, although seamlessly implementable with Jax, results in longer simulation times due to the 64 bit-precision requirement (32 bit-precision for real and imaginary parts respectively).

Supplementary References

[S1] Walter Appel. Mathematics for physics and physicists. 2007.

[S2] Henri Cartan. *Théorie élémentaire des fonctions analytiques d'une ou plusieurs varibales complexes: Avec le concours de Reiji Takahashi*. Hermann, 1961.

[S3] Maxence Ernoult, Julie Grollier, Damien Querlioz, Yoshua Bengio, and Benjamin Scellier. Updates of equilibrium prop match gradients of backprop through time in an rnn with static input. *Advances in neural information processing systems*, 32, 2019.

[S4] Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

[S5] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[S6] A Stergiou, R Poppe, and G Kalliatakis. Refining activation downsampling with softpool. arxiv 2021. *arXiv preprint arXiv:2101.00440*.

[S7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.