ARTICLE **Communicated by João Sacramento**

Frequency Propagation: Multimechanism Learning in Nonlinear Physical Networks

Vidyesh Rao Anisetti
vvaniset@syr.edu
Physics Department, Syracuse University, Syracuse, NY 13244 U.S.A.

Ananth Kandala
an.kandala@ufl.edu
Department of Physics, University of Florida, Gainesville, FL 32611, U.S.A.

Benjamin Scellier
benjamin.scellier@polytechnique.edu
Department of Mathematics, ETH Zürich, 8092 Zürich, Switzerland

J. M. Schwarz
jmschw02@syr.edu
Physics Department, Syracuse University, Syracuse, NY 13244 U.S.A., and Indian Creek Farm, Ithaca, NY 14850, U.S.A.

We introduce frequency propagation, a learning algorithm for nonlinear physical networks. In a resistive electrical circuit with variable resistors, an activation current is applied at a set of input nodes at one frequency and an error current is applied at a set of output nodes at another frequency. The voltage response of the circuit to these boundary currents is the superposition of an activation signal and an error signal whose coefficients can be read in different frequencies of the frequency domain. Each conductance is updated proportionally to the product of the two coefficients. The learning rule is local and proved to perform gradient descent on a loss function. We argue that frequency propagation is an instance of a multimechanism learning strategy for physical networks, be it resistive, elastic, or flow networks. Multimechanism learning strategies incorporate at least two physical quantities, potentially governed by independent physical mechanisms, to act as activation and error signals in the training process. Locally available information about these two signals is then used to update the trainable parameters to perform gradient descent. We demonstrate how earlier work implementing learning via chemical signaling in flow networks (Anisetti, Scellier, et al., 2023) also falls under the rubric of multimechanism learning.

Neural Computation **36, 596–620** (2024)
https://doi.org/10.1162/neco_a_01648

1 Introduction

Advancements in artificial neural networks (ANN; Goodfellow et al., 2016) have inspired a search for adaptive physical networks that can be optimized to achieve desired functionality (Anisetti, Scellier et al., 2023; Kendall et al., 2020; Stern et al., 2020, 2021; Lopez-Pastor & Marquardt, 2021; Dillavou et al., 2021; Scellier et al., 2022; Stern & Murugan, 2022). Similar to ANNs, adaptive physical networks modify their learning degrees of freedom to approximate a desired input-to-output function, but unlike ANNs, they achieve this using physical laws. In a physical network, the input is typically an externally applied boundary condition, and the output is the network's response to this input or a statistic of this response. For instance, in a resistive network, an input signal can be fed in the form of applied currents or voltages, and the output may be the vector of voltages across a subset of nodes of the network. The learning degrees of freedom of the network are, for example, the conductances of the resistors (assuming variable resistors). Ideally, these learning parameters must be updated using only locally available information. Otherwise, the network would require additional channels to transmit the gradient information.

Moreover, these parameter updates should preferably follow the direction of gradient descent in the loss function landscape, as is the case for ANNs.

Existing learning algorithms for adaptive physical networks include equilibrium propagation (Scellier & Bengio, 2017; Kendall et al., 2020) and coupled learning (Stern et al., 2021). These algorithms are based on the idea of contrastive learning (Baldi & Pineda, 1991) and proceed as follows. In a first phase, an input is presented to the network in the form of boundary currents or voltages, and the network is allowed to settle to equilibrium (the free state), where a supervisor checks the output of the system. Then the supervisor nudges the output toward the desired output. This perturbation causes the system to settle to a new (perturbed) equilibrium state, which is a slightly more accurate approximation of the function that one wants to learn. The supervisor then compares the perturbed state with the free state to make changes in the learning degrees of freedom in such a way that the network spontaneously produces an output that is slightly closer to the desired output. In the limit of infinitesimal nudging, this procedure performs gradient descent on the squared prediction error (Scellier & Bengio, 2017).

The described procedure presents practical challenges when applied to physical systems. Specifically, it requires either the retention of the network's free state for comparison with the perturbed state or the utilization of an additional, duplicate network for transmitting supplementary signals. For example, in their experimental work, Dillavou et al. (2021) use two copies of the same network to compute the free and perturbed states. Alternatively, Yi et al. (2023) use a single network, but they make use of

additional external memory to store the two states before performing the weight updates. Another idea, proposed by Kendall et al. (2020), is to use a capacitor (sample-and-hold amplifier) at each node to store the free state values, but this idea has not been verified experimentally. In this work, we propose an alternative multimechanism learning approach to overcome this hurdle. Our approach incorporates two physical quantities, each driven by its own respective mechanisms: one quantity acting as an activation signal and the other acting as an error signal. This concept is motivated by biological systems implementing functionality via multiple biophysical routes or mechanisms. Such functionality can be chemical, electrical, or even mechanical in nature with potential interactions between such mechanisms. For instance, in the brain, activity can propagate from one cell to another via electrical and chemical synapses, as opposed to just one mechanism (Pereda, 2014). Given this modular functionality in biology, it would be remiss not to explore such richness in how adaptive physical networks learn. Alternatively, as we shall soon see, this modularity is not necessarily in terms of mechanical versus chemical versus electrical signals but distinguishable signals.

We introduce frequency propagation (Freq-prop), a physical learning algorithm falling under the umbrella concept of multimechanism learning. In Freq-prop, the activation and error signals are both sent through a single channel but are encoded in different frequencies of the frequency domain; we can thus obtain the respective responses of the system through frequency (Fourier) decomposition. This algorithm, which we show to perform gradient descent, can be used to train adaptive nonlinear networks such as resistor networks, elastic networks, and flow networks. Freq-prop thus has the potential to be an all-encompassing approach. (See Figure 1 for a graphical summary of Freq-prop). In the next section, we present this idea of frequency propagation in the context of resistor networks, and in section 3, we show that frequency propagation is an example of multimechanism learning and can be generalized to train various physical systems like flow and mechanical networks. In section 4, we demonstrate this idea by training linear resistor networks to classify the Iris dataset (Fisher, 1988).

2 Nonlinear Resistive Networks

A resistive network is an electrical circuit of nodes interconnected by resistive devices, which includes linear resistors and diodes. Let N be the number of nodes in the network, and denote v_j the electric potential of node j. A subset of the nodes are input nodes, where we can set input currents: we denote x_j the input current at input node j. For each pair of nodes j and k, we denote θ_{jk} the conductance of the resistor between these nodes (provided that the corresponding branch contains a linear resistor). We further denote $\theta = \{\theta_{jk}\text{: linear branch } (j,k)\}$ the vector of conductances

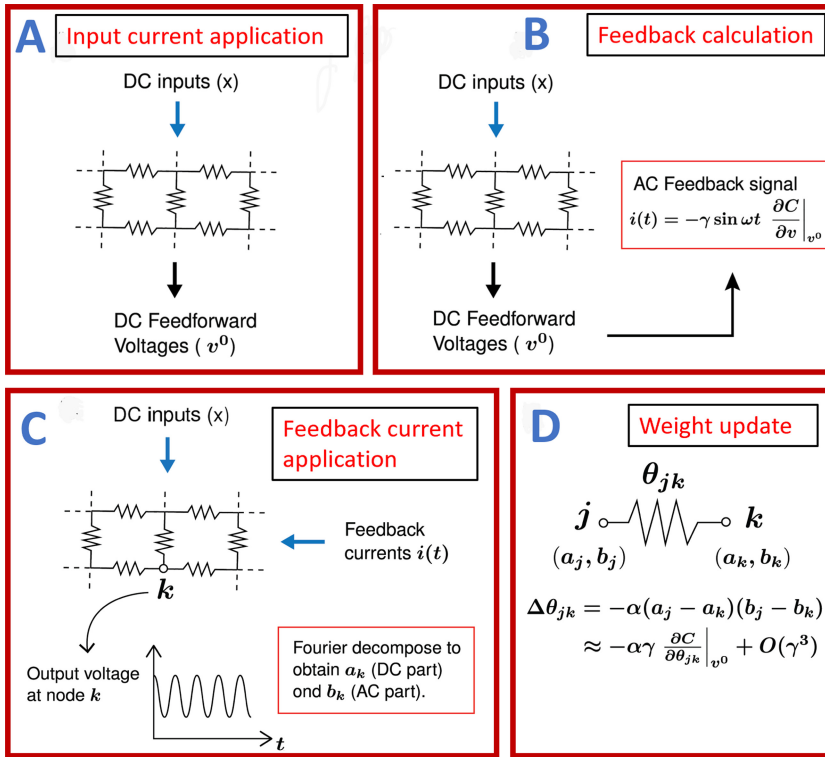

Figure 1. Illustration of the multistep weight update process in a physical network: (A) DC inputs (x) are applied to the network, resulting in a baseline voltage response (v^0). (B) Calculation of an AC feedback signal based on the derivative of the cost function with respect to the output voltages at equilibrium. (C) Application of the AC feedback signal to the network, with voltage responses at each node recorded over time. Fourier decomposition extracts the DC and AC components (a_k) and (b_k). (D) Adjustment of network weights (θ_{jk}) using the DC and AC components, adhering to the learning rule to perform gradient descent on the cost function.

and $x = (x_1, x_2, \ldots, x_N)$ the vector of input currents, where by convention, $x_j = 0$ if node j is not an input node. Finally, we denote $v = (v_1, v_2, \ldots, v_N)$ the configuration of the nodes' electric potentials and $v(\theta, x)$ the equilibrium value of v as a function of the branch conductances (θ) and the input currents (x).

The following result, known since the work of Millar (1951), provides a characterization of the equilibrium state—see also Kendall et al. (2020) for a proof of this result with notations closer to ours.

Theorem 1. *There exists a real-valued function $E(\theta, x, v)$ such that*

$$v(\theta, x) = \arg\min_{v} \; E(\theta, x, v). \tag{2.1}$$

Furthermore, $E(\theta, x, v)$ is of the form

$$E(\theta, x, v) = E_{\text{input}}(x, v) + E_{\text{nonlinear}}(v) + \sum_{\text{linear branch } (j,k)} \frac{1}{2}\theta_{jk}\left(v_j - v_k\right)^2, \tag{2.2}$$

where $E_{\text{input}}(x, v)$ is a function of x and v and $E_{\text{nonlinear}}(v)$ is a function of v only.

$E(\theta, x, v)$ is the energy function of the system, also called the co-content (Millar, 1951), and the equilibrium state $v(\theta, x)$ is a minimizer of the energy function. The energy function contains an energy term $E_{\text{input}}(x, v)$ associated with boundary input currents x. It also contains energy terms of the form $\theta_{jk}(v_j - v_k)^2$ representing the power dissipated in branch (j, k). The term $E_{\text{nonlinear}}(v)$ contains all nonlinearities of the system. In a linear resistance network (i.e., when $E_{\text{nonlinear}}(v) = 0$), it is well known that the equilibrium configuration of node electric potentials minimizes the power dissipation. Theorem 1 generalizes this result to nonlinear networks. Below we explain how the different terms of $E(\theta, x, v)$ are constructed.

2.1 Constructing the Energy Function. Each branch is characterized by its current-voltage characteristic, $i_{jk} = \widehat{i}_{jk}(v_j - v_k)$, where $\widehat{i}_{jk}(\cdot)$ is a real-valued function that returns i_{jk}, the current flowing from j to k in response to the voltage $v_j - v_k$. The energy term corresponding to branch (j, k), called the co-content of the branch (Millar, 1951), is, by definition,

$$E_{jk}(v_j - v_k) = \int_0^{v_j - v_k} \widehat{i}_{jk}(v')dv'. \tag{2.3}$$

In general, the characteristic function $\widehat{i}_{jk}(\cdot)$ is arbitrary, (i.e., nonlinear). However, some branches are linear, meaning that their current-voltage characteristic is of the form $i_{jk} = \theta_{jk}\left(v_j - v_k\right)$, where θ_{jk} is the branch conductance.[1] For such linear branches, the energy term is

$$E_{jk}(v_j - v_k) = \frac{1}{2}\theta_{jk}\left(v_j - v_k\right)^2, \tag{2.4}$$

which is the power dissipated in branch (j, k).

[1] To avoid any confusion, we stress that θ_{jk} is a scalar, whereas $\hat{i}_{jk}(\cdot)$ is a real-valued function. Thus, $\theta_{jk}\left(v_j - v_k\right)$ denotes the product of θ_{jk} and $v_j - v_k$, whereas $\hat{i}_{jk}(v_j - v_k)$ denotes the function $\hat{i}_{jk}$ applied to the voltage $v_j - v_k$.

We gather all the energy terms of nonlinear branches under a unique term,

$$E_{\text{nonlinear}}(v) = \sum_{\text{nonlinear branch } (j,k)} E_{jk}(v_j - v_k), \tag{2.5}$$

where we recall that $v = (v_1, v_2, \ldots, v_N)$.

As for the energy term $E_{\text{input}}(x, v)$, we present two ways to impose boundary conditions to the network to feed it with input signals x in the form of boundary currents or boundary electric potentials. Recall that we write $x = (x_1, x_2, \ldots, x_N)$, the vector of input signals, where $x_j = 0$ if node j is not an input node. In the case of boundary currents, the corresponding energy term is

$$E^{\text{current}}_{\text{input}}(x, v) = \sum_{j \in \{\text{input nodes}\}} x_j v_j, \tag{2.6}$$

whereas in the case of boundary electric potentials, the energy term is

$$E^{\text{voltage}}_{\text{input}}(x, v) = \begin{cases} 0 & \text{if} \quad v_j = x_j, \ \forall j \in \{\text{input nodes}\}, \\ +\infty & \text{otherwise}, \end{cases} \tag{2.7}$$

that is, the electric potential v_j is clamped to x_j for every input node j (so that the energy remains finite).

Putting all the energy terms together and denoting $E_{\text{input}}(x, v)$, the energy term of input signals (either $E^{\text{current}}_{\text{input}}(x, v)$ or $E^{\text{voltage}}_{\text{input}}(x, v)$ depending on the case), we get the energy function of equation 2.2.

3 Multimechanism Learning via Frequency Propagation

Learning in a resistive network consists in adjusting the branch conductances (θ) so that the network exhibits a desired behavior: an input-output function $x \mapsto v(\theta, x)$. In machine learning, this problem is formalized by introducing a cost function C. Given an input-output pair (x, y), the quantity $C(v(\theta, x), y)$ measures the discrepancy between the model prediction $v(\theta, x)$ and the desired output y. The learning objective is to find the parameters θ that minimize the expected cost $\mathbb{E}_{(x,y)}\big[C(v(\theta, x), y)\big]$ over input-output pairs (x, y) coming from the data distribution for the task that the system must solve.

In deep learning, the main tool for this optimization problem is stochastic gradient descent (SGD; Bottou, 2010). At each step, we pick at random an example (x, y) from the training set and update the parameters as

$$\Delta\theta = -\eta \frac{\partial \mathcal{L}}{\partial \theta}(\theta, x, y), \tag{3.1}$$

where η is a step size and

$$\mathcal{L}(\theta, x, y) = C(v(\theta, x), y) \tag{3.2}$$

is the loss function.

We now present frequency propagation (Freq-prop), a learning algorithm for physical networks whose update rule performs SGD. Freq-prop proceeds by modifying the energy of the network to push or pull away the network's output values from the desired outputs. In the case of a resistive network (see section 2), we inject sinusoidal currents at the output nodes of the network, $i(t) = \gamma \sin(\omega t) \frac{\partial C}{\partial v}(v, y)$, where t denotes time, ω is a frequency, and γ is a small positive constant.[2] This augments the energy function of the system by a time-dependent sinusoidal energy term $\gamma \sin(\omega t) C(v, y)$. Due to this perturbation, the system's response $v(t)$ minimizing the energy at time t is

$$v(t) = \arg\min_{v} \left[E(\theta, x, v) + \gamma \sin(\omega t) C(v, y) \right]. \tag{3.3}$$

The response $v(t)$ is periodic of period $T = 2\pi/\omega$, and for small perturbations (i.e., $\gamma \ll 1$), it is approximately sinusoidal. It is important to note that we have assumed that the system equilibrates at timescales much smaller than the time period of the error signal. Without this rapid equilibration, the system's response will exhibit a noticeable delay compared to the instantaneous response predicted by equation 3.3. Next, we assume that we can recover the first two vectors of Fourier coefficients of $v(t)$, that is, the vectors a and b, such that

$$a = \frac{1}{T} \int_0^T v(t) dt, \qquad b = \frac{2}{T} \int_0^T v(t) \sin(\omega t) dt. \tag{3.4}$$

Finally, denoting $a = (a_1, a_2, \ldots, a_N)$ and $b = (b_1, b_2, \ldots, b_N)$, we update each parameter θ_{jk} according to the learning rule

$$\Delta\theta_{jk} = -\alpha(b_j - b_k) \cdot (a_j - a_k), \tag{3.5}$$

where α is a positive constant.

[2] In practical situations such as the squared error prediction, the cost function C depends only on the state of output nodes; therefore, nudging requires injecting currents at output nodes only.

Theorem 2. *For every parameter θ_{jk}, we have*

$$\Delta\theta_{jk} = -\alpha\,\gamma\,\frac{\partial \mathcal{L}}{\partial \theta_{jk}}(\theta, x, y) + O(\gamma^3) \tag{3.6}$$

when $\gamma \to 0$.

Namely, the learning rule 3.5 approximates one step of gradient descent with respect to the loss, with learning rate $\alpha\,\gamma$. Note that this learning rule is local: it requires solely locally available information for each parameter θ_{jk}.

Proof. Let θ, x, and y be fixed. For every $\beta \in \mathbb{R}$, we denote

$$v_\star^\beta = \arg\min_v \left[E(\theta, x, v) + \beta\, C(v, y)\right]. \tag{3.7}$$

With this notation, note that the response $v(t)$ of equation 3.3 rewrites $v(t) = v_\star^{\gamma \sin(\omega t)}$. Let us write the second-order Taylor expansion of $v_\star^\beta$ around $\beta = 0$:

$$v_\star^\beta = v_\star^0 + \beta\,\left.\frac{\partial v_\star^\beta}{\partial \beta}\right|_{\beta=0} + \frac{\beta^2}{2}\,\left.\frac{\partial^2 v_\star^\beta}{\partial \beta^2}\right|_{\beta=0} + O(\beta^3), \tag{3.8}$$

where $v_\star^0 = v(\theta, x)$ by equation 2.1, and $\left.\frac{\partial v_\star^\beta}{\partial \beta}\right|_{\beta=0}$ and $\left.\frac{\partial^2 v_\star^\beta}{\partial \beta^2}\right|_{\beta=0}$ denote the derivative and second derivative of $v_\star^\beta$ at $\beta = 0$. Taking $\beta = \gamma \sin(\omega t)$ in the above formula, we get

$$v(t) = v_\star^{\gamma \sin(\omega t)} = v_\star^0 + \gamma \sin(\omega t)\,\left.\frac{\partial v_\star^\beta}{\partial \beta}\right|_{\beta=0} \tag{3.9}$$

$$+ \frac{\gamma^2}{2}\sin(\omega t)^2\,\left.\frac{\partial^2 v_\star^\beta}{\partial \beta^2}\right|_{\beta=0} + O(\gamma^3) \tag{3.10}$$

uniformly in t. Therefore, the first two vectors of Fourier coefficients a and b of the periodic function $v(t)$, with time period $T = 2\pi/\omega$ are

$$a = \frac{1}{T}\int_0^T v(t)dt = v_\star^0 + \frac{\gamma^2}{4}\,\left.\frac{\partial^2 v_\star^\beta}{\partial \beta^2}\right|_{\beta=0} + O(\gamma^3), \tag{3.11}$$

$$b = \frac{2}{T}\int_0^T v(t)\sin(\omega t)dt = \gamma\,\left.\frac{\partial v_\star^\beta}{\partial \beta}\right|_{\beta=0} + O(\gamma^3). \tag{3.12}$$

Next, we know from the equilibrium propagation formula (see theorem 2.1 in Scellier, 2021) that the gradient of the loss $\mathcal{L}$ is equal to

$$\frac{\partial \mathcal{L}}{\partial \theta}(\theta, x, y) = \left.\frac{d}{d\beta}\right|_{\beta=0} \frac{\partial E}{\partial \theta}(\theta, x, v_\star^\beta). \tag{3.13}$$

Therefore,

$$\frac{\partial \mathcal{L}}{\partial \theta}(\theta, x, y) = \frac{\partial^2 E}{\partial \theta \partial v}(\theta, x, v_\star^0) \cdot \left.\frac{\partial v_\star^\beta}{\partial \beta}\right|_{\beta=0}. \tag{3.14}$$

Multiplying both sides by γ and using equation 3.12, we get

$$\gamma \frac{\partial \mathcal{L}}{\partial \theta}(\theta, x, y) = \frac{\partial^2 E}{\partial \theta \partial v}(\theta, x, v_\star^0) \cdot b + O(\gamma^3). \tag{3.15}$$

Finally, given the form of the energy function (equation 2.2) and using $b = O(\gamma)$ and $v_\star^0 = a + O(\gamma^2)$ from equation 3.11, we get for every parameter θ_{jk},

$$\gamma \frac{\partial \mathcal{L}}{\partial \theta_{jk}}(\theta, x, y) = \big(a_j - a_k\big) \cdot \big(b_j - b_k\big) + O(\gamma^3). \tag{3.16}$$

Therefore, the learning rule

$$\Delta\theta_{jk} = -\alpha (b_j - b_k) \cdot (a_j - a_k) \tag{3.17}$$

satisfies

$$\Delta\theta_{jk} = -\alpha\, \gamma \frac{\partial \mathcal{L}}{\partial \theta_{jk}}(\theta, x, y) + O(\gamma^3). \tag{3.18}$$

Hence the result. □

Remark 1. For simplicity, we have omitted the time of relaxation to equilibrium in our analysis. However, a practical circuit has an effective capacitance C_{eff} and therefore will equilibrate in time $\tau_{\text{relax}} \sim R_{\text{eff}} C_{\text{eff}}$, where R_{eff} is the effective resistance of the circuit. Our learning algorithm will work as long as the circuit equilibrates much faster than the timescale of oscillation ($\tau_{\text{relax}} \ll 1/\omega$). Our analysis thus requires that C_{eff} be small enough for the assumption $\tau_{\text{relax}} \ll 1/\omega$ to hold.

If this is not the case, there will be a trade-off between how fast one can train the network with Freq-Prop versus how accurate the approximation

is for gradient descent. We leave the study of the regime where C_{eff} is non-negligible for future work.

We note however that the effective capacitance of the circuit is expected to grow linearly with the size of the network (the total amount of wire), so that inference time grows linearly with the size of the network too. We also note that the same is true for deep neural networks: in a feedforward network, both inference (the forward pass) and training (backpropagation) grow linearly with the size of the network.

Remark 2. While our nudging method, equation 3.3, is inspired by the one of equilibrium propagation (Scellier & Bengio, 2017; Kendall et al., 2020), it is also possible to apply the nudging variant of coupled learning (Stern et al., 2021) which might be easier to implement in practice (Dillavou et al., 2021). To do this, we denote v_O^F the "free" equilibrium value of the output nodes of the network (where the prediction is read), without nudging. Then, at time t, we clamp the output nodes to $v_O^C(t) = v_O^F + \gamma \sin(\omega t)(y - v_O^F)$. This nudging method can be achieved via AC voltage sources at output nodes. We note, however, that theorem 2 does not hold with this alternative nudging method.

Remark 3. Measuring b_j for every node j as per equation 3.4 requires that we use the same reference time $t = 0$ for all nodes, that is, it requires global synchronization of the measurements for all nodes.

However, in practice, there may be a time delay t_j between nudging and measurement, leading to a measured response $v_j(t) = a_j + b_j \sin(\omega(t + t_j)) + O(\gamma^3)$ at node j. Without any information about t_j, we can only obtain the absolute value of the coefficient b_j, not its sign.

We propose a solution to this issue in section 5.

4 Demonstration of Freq-Prop in Linear Networks

We trained a network, comprising 40 nodes and 239 edges, to learn the Iris data set (see Figure 2). Each edge in the network is assigned a conductance value, drawn from a uniform distribution $[10^{-5}, 1]$. Throughout the training process, the weights are constrained to the range $[10^{-5}, 20]$. We set the nudge amplitude, η, to 1 and the learning rate, α, to 0.01. Our training methodology predominantly aligns with the procedures outlined in our previous work (Anisetti, Scellier et al., 2023), albeit with a few modifications:

1. Input to the network is applied as a DC voltage at input nodes. The response to this input is calculated at the output nodes, and error is determined.
2. For each discrete time step, t, with a time interval $\Delta t = 0.05$ between two consecutive steps, the discrepancy between the error is modulated by multiplying it with $\sin(\omega t)$ (here, $\omega = 2\pi$, period $= 1$).

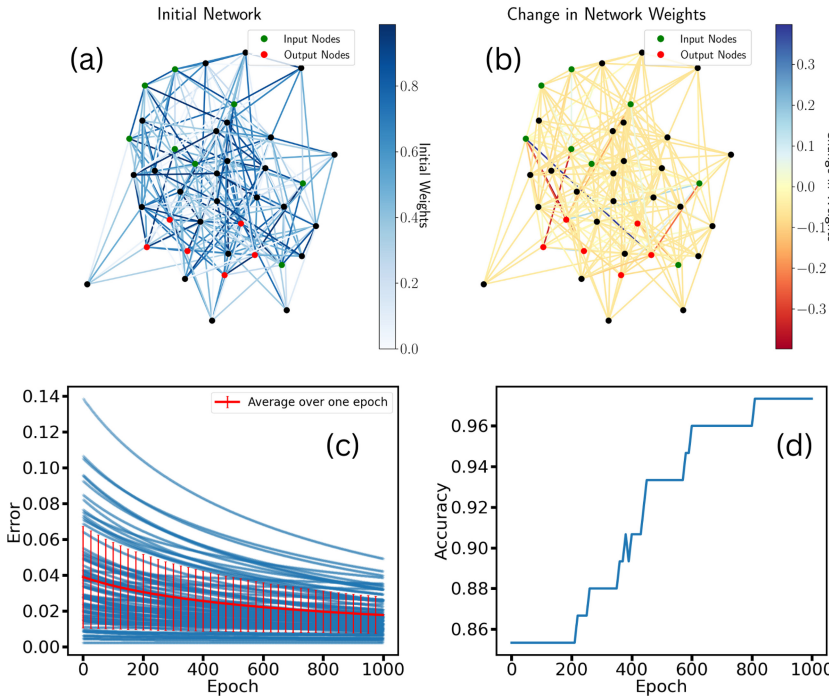

Figure 2. Training on the Iris data set: (a) The initial network, consisting of 40 nodes and 390 edges, is depicted before training; the color bar represents the conductance values. (b) The alterations in the network after 1000 epochs of training. (c) The error versus the epoch plot. The Iris data set consists of 150 examples, with 75 used for training and the remainder reserved for testing. Each epoch trains the network on 75 examples, where, at every step, an example is presented to the network, and the error between the network's output and the desired output is computed, represented by a blue dot. The red line denotes the average error. Here, error is defined as the square root of the mean squared error. (d) The Accuracy versus the epoch plot, where accuracy is quantified as the fraction of correct predictions from the total testing examples.

This AC voltage is applied over a duration of five units of time, encompassing five complete oscillation periods. This error signal is applied along with the DC signal.

3. The response of each node is recorded over time, resulting in a time series of voltage values for each node in the network. This time series contains both the steady-state (DC) component and the oscillating (AC) component of the response. Then we decompose the responses at each node into AC and DC components using Fourier series method (see equation 3.4).

4. The DC component represents the average or steady-state response of the node. It is calculated by taking the mean of the time series of the response at each node.
5. The AC component is extracted by multiplying the time series of the response by $\sin(\omega t)$ and then averaging the result.
6. Using these AC and DC components, the weights are updated (see equation 3.5).

We have detailed the entire training procedure in appendix A.

5 Choice of the Nudging Signal

We have seen in section 3 that when a sinusoidal nudging signal $\gamma \sin(\omega t) C(v, y)$ is used, the measured response at node j will be of the form $v_j(t) = a_j + b_j \sin(\omega(t + t_j)) + O(\gamma^3)$, where t_j is the time delay between nudging and measurement. Unfortunately, it is not possible to recover the sign of b_j without any knowledge of t_j. This problem can be overcome by using a different nudging signal.

In general, if we nudge the system by an energy term $\gamma f(t) C(v, y)$, where $f(t)$ is an arbitrary function such that $\sup_t |f(t)| < \infty$, then the system's response at node j will be of the form $v_j(t) = a_j + b_j f(t + t_j) + O(\gamma^2)$.

Our goal is to choose an f so that we can obtain for every node j the values of a_j and b_j by measuring only $v_j(t)$, without knowing t_j.

Clearly this is not possible for all functions f. For example, if $f(\cdot)$ is a constant, then $v_j(\cdot)$ is also a constant, and we cannot recover the values of a_j and b_j from $v_j(\cdot)$ alone. Another example for which this is not possible is $f(t) = \sin(\omega t)$. This is because a time delay $t_j = \pi/\omega$ will change the sign of the signal, $\sin(\omega(t + t_j)) = -\sin(\omega t)$; therefore, the sign of b_j cannot be recovered without any knowledge of t_j.

An example of a nudging signal for which we can infer the values of a_j and b_j (up to $O(\gamma^2)$) is $f(t) = |\sin(\omega(t))|$. To do this, we observe the response at node j,

$$v_j(t) = a_j + b_j |\sin(\omega(t + t_j))| + O(\gamma^2), \tag{5.1}$$

for a duration τ_{obs} greater than the time period of the signal $T = 2\pi/\omega$. The coefficients a_j and b_j can be obtained by identifying the times where the signal's derivative is zero or is discontinuous. Specifically, denoting $\partial_+ v_j(t)$ and $\partial_- v_j(t)$ the left and right derivatives of the signal at time t, we have

$$a_j = v_j(t_1) + O(\gamma^2) \text{ where } \partial_+ v(t_1) \neq \partial_- v(t_1), \tag{5.2}$$

$$b_j = v_j(t_2) - v_j(t_1) + O(\gamma^2) \text{ where } \partial v(t_2) = 0. \tag{5.3}$$

More generally, we show in appendix B that in principle, it is possible to recover the coefficients a_j and b_j if and only if the function f has the

property that there is no τ such that $f(t) = \sup f + \inf f - f(t+\tau)$ for every t. In other words, no amount of time delay converts the signal's upright form to its inverted form or vice versa.

6 General Applicability of Frequency Propagation

Freq-prop applies to arbitrary physical networks: not only resistive networks, but also flow networks, capacitive networks, and inductive networks, among others. In these networks, the notion of current-voltage characteristics will be replaced by current-pressure characteristics, current-flux characteristics, and charge-voltage characteristics, respectively. The mathematical framework for nonlinear elements (see section 2) also applies to these networks, where the energy functions minimized at equilibrium are the co-content, the inductive energy, and the capacitive co-energy, respectively (Millar, 1951; Cherry, 1951).

To emphasize the generality of Freq-prop, we present it here in the context of central force spring networks (or elastic networks; Stern et al., 2021), as well as Hopfield networks (the Ising model).

6.1 Central Force Spring Networks. We consider an elastic network of N nodes interconnected by springs. The elastic energy stored in the spring connecting node i to node j is $E_{ij}(r_{ij}) = \frac{1}{2}k_{ij}(r_{ij} - \ell_{ij})^2$, where k_{ij} is the spring constant, ℓ_j is the spring's length at rest, and r_{ij} is the distance between nodes i and j. Nonlinear springs are also allowed, and their energy terms are gathered in a unique term, $E_{\text{nonlinear}}(r)$. Thus, the total elastic energy stored in the network, which is minimized, is given by

$$E(\theta, r) = \frac{1}{2}\sum_{i,j} k_{ij}\left(r_{ij} - \ell_{ij}\right)^2 + E_{\text{nonlinear}}(r), \tag{6.1}$$

where $\theta = \{k_{ij}, \ell_{ij}\}$ is the set of adjustable parameters and $r = \{r_{ij}\}$ plays the role of state variable.

In this setting, as in the case of resistive networks, we apply a nudging signal $\gamma \sin(\omega t) C(r, y)$ at the output part of the network, we observe the response $r(t)$, and we assume that we can recover the first two vectors of Fourier coefficients of $r(t)$, that is, the vectors a and b such that $a = \frac{1}{T}\int_0^T r(t)dt$ and $b = \frac{2}{T}\int_0^T r(t)\sin(\omega t)dt$. Then the learning rules for the spring constant k_{ij} and the spring's length at rest ℓ_{ij} read, in this context,

$$\Delta k_{ij} = -\alpha\, b_{ij}\,(a_{ij} - \ell_{ij}), \qquad \Delta \ell_{ij} = -\alpha\, k_{ij}\, b_{ij}. \tag{6.2}$$

Theorem 2 generalizes to this setting; the above learning rules perform stochastic gradient descent on the loss: $\Delta\theta = -\alpha\gamma\frac{\partial \mathcal{L}}{\partial \theta}(\theta, x, y) + O(\gamma^3)$.

6.2 Continuous Hopfield Networks. Freq-prop also applies to Hopfield networks (the Ising model; Hopfield, 1984; Baldi & Pineda, 1991). In a Hopfield network of multiple units interconnected by synapses, the energy term between unit i and unit j is $E_{ij} = w_{ij}h_ih_j$, where w_{ij} is the synaptic weight and h_i is the state of unit i.

The total energy is

$$E(\theta, h) = -\sum_{i,j} w_{ij}h_ih_j, \tag{6.3}$$

where $\theta = \{w_{ij}\}$ is the set of adjustable parameters and $h = \{h_i\}$ plays the role of state variable. After applying a nudging signal $\gamma \sin(\omega t)C(h, y)$ at a set of output units, we observe the response $u(t)$ (the state of the units at equilibrium), and we compute the vectors a and b such that $a = \frac{1}{T}\int_0^T u(t)dt$ and $b = \frac{2}{T}\int_0^T u(t)\sin(\omega t)dt$. The learning rules for the weight w_{ij} read

$$\Delta w_{ij} = -\alpha(a_ib_j + a_jb_i), \tag{6.4}$$

which performs stochastic gradient descent on the loss, up to $O(\gamma^3)$.

7 Related Work

Frequency propagation builds on learning via chemical signaling (Anisetti, Scellier et al., 2023), another example of multimechanism learning (MmL) in physical networks. Whereas MmL via frequency propagation uses two different frequencies to play the role of the activation and error signals during training, MmL via chemical signaling uses two different chemical concentrations for these signals. Anisetti, Scellier et al. (2023) present learning via chemical signaling in the setting of linear flow networks, which we extend here to the nonlinear setting (see appendix C).

Freq-prop is also related to equilibrium propagation (EP; Scellier & Bengio, 2017; Kendall et al., 2020) and coupled learning (Stern et al., 2021). To see the relationship with these algorithms, we consider the case of resistive networks (see section 2). Denote $v_{jk} = v_j - v_k$ the voltage across branch (j, k). Further denote $v^\beta = \arg\min_v \left[E(\theta, x, v) + \beta C(v, y)\right]$ for any $\beta \in \mathbb{R}$. Kendall et al. (2020) proved based on a result from Scellier & Bengio (2017) that the learning rule

$$\Delta^{\text{EP}}\theta_{jk} = \frac{\alpha}{2}\left((v^0_{jk})^2 - (v^\beta_{jk})^2\right) \tag{7.1}$$

performs gradient descent with step size $\alpha\beta$, up to $O(\beta^2)$. We note that the right-hand side of equation 7.1 is also equal to $\alpha\, v^0_{jk}\left(v^0_{jk} - v^\beta_{jk}\right) + O(\beta^2)$, showing that the gradient information is contained in the physical

quantities v^0 and $\frac{\partial v^\beta}{\partial \beta}\big|_{\beta=0}$. These quantities correspond to the activation and error signals of Freq-prop, respectively. To avoid the use of finite differences to measure $\frac{\partial v^\beta}{\partial \beta}\big|_{\beta=0}$, Freq-prop makes use of a time-varying nudging signal $\beta(t) = \gamma \sin(\omega t)$. With this method, the activation and error signals are encoded in the frequencies 0 and ω of the response signal $v(t) = v^0 + \gamma \sin(\omega t)\frac{\partial v^\beta}{\partial \beta}\big|_{\beta=0} + O(\gamma^3)$. The required information can thus be recovered via frequency analysis.

The idea of using an oscillating nudging signal was also proposed by Baldi and Pineda (1991) and more recently (concurrent with our work) in holomorphic EP (Laborieux & Zenke, 2022). Our work differs from these two other works in several ways. First, our learning rule can be decomposed as activation signal times error signal ($a \times b$), whereas the learning rule of Baldi and Pineda (1991) takes the form $\Delta\theta_{jk} = \frac{\alpha}{2} \int \sin(\omega t) v_{jk}(t)^2 dt$, and similarly for holomorphic EP. Second, our learning rule is proved to approximate the gradient of the cost function, up to $O(\beta^3)$, unlike in Baldi and Pineda (1991). Laborieux and Zenke (2022) exploit the Cauchy formula of complex calculus to prove that their algorithm computes the exact gradient of the cost function, independent of the strength of the nudging signal. To achieve this, the authors allow the nudging factor to take complex values, $\beta = \gamma e^{i\omega t} \in \mathbb{C}$, and the domain of definition of the energy function $v \mapsto E(\theta, x, v)$ is extended to complex configurations $v \in \mathbb{C}^N$. However, it is not yet straightforward to see how this mathematical formalism can be directly mapped to physical systems such as resistive networks or spring networks, the motivation of our work.

Another recent work (McCaughan et al., 2023) presents a technique for gradient computation in analog DNNs. The authors modulate the parameters, θ_i, using periodic functions with distinct frequencies, ω_i. This results in the cost function oscillating as a superposition of these frequencies. They demonstrated that the oscillation amplitude of the cost function carries gradient information. The gradient with respect to individual weights, θ_i, can be discerned by isolating the Fourier component of the cost function oscillating at frequency ω_i. While this approach shares the concept of signal decomposition with our frequency propagation method, there are key differences. In our approach, we modulate the output error using a periodic function, leading to an AC feedback signal and a DC feedforward signal. We then extract individual signals through Fourier decomposition.

Another recent work proposes an alternative approach to train physical systems by gradient descent, agnostic equilibrium propagation (Scellier et al., 2022). However, this method imposes constraints on the nature of the parameters (θ), which must minimize the system's energy (E), just like the state variables (v) do. This assumption does not allow us to view the conductances of resistors (resp. the spring constants) as trainable parameters in a resistive network (resp. in a spring network). The method also requires control knobs with the ability to perform homeostatic control over

the parameters. Our work can also be seen as a physical implementation of implicit differentiation in physical networks. We refer to Zucchet and Sacramento (2022) for a description of implicit differentiation where the authors use a mathematical formalism close to ours.

Finally, other physical learning algorithms that make explicit use of time are being developed. For instance, recent work proposes a way to train physical systems with time-reversible Hamiltonians (Lopez-Pastor & Marquardt, 2021). In this method, called *Hamiltonian echo backpropagation* (HEB), the error signal is a time-reversed version (an echo) of the activation signal, with the cost function acting as a perturbation on this signal.

However, HEB requires a feasible way to time-reverse the activation signal.

8 Discussion

We have introduced frequency propagation (Freq-prop), a physical learning algorithm that falls in the category of multimechanism learning (MmL). In MmL, separate and distinguishable activation and error signals contribute to a local learning rule, such that trainable parameters (e.g., conductances of variable resistors) perform gradient descent on a loss function.

In Freq-prop, the activation and error signals are implemented using different frequencies of a single physical quantity (e.g., voltages or currents) and are thus distinguishable. We note however that the distinguishability of the signals does not mean that they are mathematically independent. In Freq-prop, the error signal depends on the activation signal via the Hessian of the network.

Other potential MmL algorithms may involve independent physical mechanisms, such as an electrical activation signal and a chemical error signal or vice versa. Multimechanism learning algorithms, such as Freq-prop, may have implications toward designing fast and low-power, or high-efficiency, hardware for AI, as they are rooted in physical principles. For the time being, inroads are being made by using backpropagation to train controllable physical systems in a hybrid in silico–in situ approach (Wright et al., 2022). As we work toward a fully in situ approach, Freq-prop is a natural candidate. And while the in situ realization of a nonlinear resistor network is an obvious starting point, there are potential limitations, particularly in terms of timescales. Consider the time of relaxation to equilibrium (τ_{relax}), the timescale of the sinusoidal nudging signal ($T = 2\pi/\omega$), and the timescale of learning (τ_{learning}). Our learning methodology requires that $\tau_{\text{relax}} \ll T < \tau_{\text{learning}}$. More specifically:

1. Once input is applied, the network reaches equilibrium in time τ_{relax}.
2. Based on the network's output, a sinusoidal nudging signal of frequency ω is applied at the output nodes. The timescale of evolution of

this sinusoidal nudging wave is $T = 2\pi/\omega$. Assuming that $\tau_{\text{relax}} \ll T$, the network is at equilibrium at every instant t.

3. We observe the network's response $v(t)$ for a time $\tau_{\text{obs}} > T$ to extract the coefficients a and b of equation 3.4. Updating the conductances of the resistors takes a time $\tau_{\text{learning}} \sim \tau_{\text{obs}}$ using the values of a and b to determine the magnitude and sign of these updates.

In our discussion of physical learning mechanisms, it's important to clarify the nature of the physical processes we envision for self-updating weights. In seeking an ideal physical mechanism for neuromorphic systems, we prioritize simplicity and intrinsic capability in each network component. The goal is to avoid complex, artificially induced modifications for self-updating capabilities. For instance, consider the contrast between a modified sliding rheostat, where weight adjustment is motorcontrolled, and a memristor. The latter represents a more straightforward approach to achieving self-updating resistance. A memristor exemplifies the kind of simplicity we are advocating for. This simplicity is not just a matter of ease of implementation, but also crucial for scalability and robustness in neuromorphic computing. Simplified components are more conducive to building larger and more efficient neural networks. They also tend to be less prone to errors compared to more complex devices like motorcontrolled rheostats. Our research, as highlighted in Anisetti, Kandala et al. (2023), is an ongoing effort to explore and develop such inherent physical mechanisms.

There are many examples of biological systems implementing functionality via multiple biophysical routes, since brains developed rather recently if one looks at the evolutionary tree dating back to the origins of life. For instance, cancer cells learn to move through mazes more efficiently using self-generated chemotaxis (Tweedy et al., 2020). Slime mold, a single-cell organism, stores memory and can learn through a combination of chemical and mechanical signals (Boussard et al., 2021). Even at the subcellular scale, chromatin can be viewed as a mechanical computer, similar to the analytical machine.

Finally, could something like Freq-prop occur in the brain? Earlier work analyzing local field potentials recorded simultaneously from different regions in the cortex suggested that feedforward signaling is carried by gamma-band (30–80 Hz) activity, whereas feedback signaling is mediated by alpha-(5–15 Hz) or beta- (14–18 Hz) band activity, though local field potentials are not actively relayed between regions (Bastos et al., 2015). More recent work in the visual cortex argues that feedforward and feedback signaling rely on separate "channels" since correlations in neuronal population activity patterns, which are actively relayed between regions, are distinct during feedforward- and feedback-dominated periods (Semedo et al., 2022). Freq-prop is also related in spirit to the idea of frequency multiplexing in biological neural networks (Naud & Sprekeler, 2018; Payeur et al.,

2021; Akam & Kullmann, 2014), which uses the simultaneous encoding of two or more signals.

While Freq-prop here uses only two separate signals—an activation signal and an error signal—one can envision multiple activation and error signals being encoded to accommodate vector inputs and outputs and to accommodate multiple, competing cost functions. With multiple activation and error signals, one can also envision coupling learning via chemical signaling (see appendix C) with Freq-prop, for example, to begin to capture the full computational creativity of the brain.

Appendix A: Details of the Training Process

We train the network on a standard machine learning task: classifying Iris flowers. The Iris data set (Fisher, 1988) contains 150 examples of Iris flowers belonging to three species (Setosa, Virginica, and Versicolor), and, therefore, 50 examples for each category. Each example is of the form $X = (X_1, X_2, X_3, X_4)$, composed of four features of the flower (petal width, petal length, sepal width, and sepal length, all measured in centimeters), and comes with its assigned Iris category, denoted Y. So an example would look something like $X = (5.1, 3.5, 1.4, 0.2)$ and $Y =$ setosa. Given the four features of the flower as input, the trained network should be able to tell which species it belongs to:

1. **Network generation:** A network consisting of N nodes, with a varying number of edges M, is created. For this, we first create a Barabási-Albert network with connection parameter 1. This graph generation algorithm connects a new node with one existing node in a manner that nodes with higher degree have a stronger likelihood for selection. This creates a network with N nodes and $N - 1$ edges. To create a network with M edges, we add $M - (N + 1)$ unique edges.
2. Four pairs of input nodes are chosen from this flow network, where the input data (the normalized features of the Iris) are given as external currents across these four pairs of boundary nodes. Three pairs of nodes are chosen as the output nodes. Once the network is trained for a given input, the set of potential drops across these output node pairs should tell the category of Iris the input data correspond to.

The network architecture remains fixed throughout the training-testing process. Only the conductances of these weights are modified. As for how the flow network interfaces with the Iris data set:

1. The data set is divided into two subsets: one training set (used for training) and one test set (used for testing). Each of these sets has 75 examples of Irises, 25 from each category.
2. The data set is normalized. That is, for each example X in the data set and for each feature X_i of X, we set $X_i^{\text{norm}} = \lambda \cdot \frac{X_i - X_i^{\min}}{X_i^{\max} - X_i^{\min}}$, where $X_i^{\max}$

and $X_i^{\min}$ are the minimum and maximum values for that feature X_i in the training set. We choose $\lambda = 5$ for all simulations.

While choosing the desired outputs, we must keep in mind the fact that this linear system may not be able to find a set of weights that give out the desired output. In other words, we must choose desired outputs that are physically attainable. We therefore implement the same technique as described by Dillavou et al. (2021) to choose the desired output voltages for each of the three Iris categories. For each category, the desired voltage is the average, normalized input data. That is, each Iris category has 25 examples of four input features, and each input feature is averaged out over 25 examples. Finally, we obtain a four-tuple of averaged input features for each Iris category. When this is given as input to the initial network, we aim to arrive at an output voltage that corresponds to the average behavior of the input, which is the desired voltage.

To conduct the training process, first the input data are given to the network as a DC voltage and the output is observed. If the output is not equal to the desired output for that Iris category, feedback AC currents are applied at the output nodes. The weights of the network are modified using the learning rule mentioned above. This process is repeated consecutively for all examples.

Once the entire data set is exhausted, we say that one epoch has passed. We train the network for multiple epochs. At the beginning of each epoch, because the network has changed significantly, new desired voltages are calculated. Therefore, each epoch has its own set of desired voltages. To conduct the testing process, after the network is trained for multiple epochs, we record how well it classifies unseen data from the test set. To be specific, the test set has 25 examples per Iris category. The Iris features from these test examples are given as input, and the output voltage is compared with the desired voltage. The desired voltage is calculated using the testing data set. An example is then classified into that Iris category, for which the output voltage is closest to the desired output.

Appendix B: Further Details on the Nudging Signal

Let $f(t)$ denote the nudging signal. Assuming that f is bounded, recall that for every j, the measured response $v_j(t)$ at node j is of the form $v_j(t) = a_j + b_j f(t + t_j) + O(\gamma^2)$, where a_j and b_j are the numbers that we wish to recover (up to $O(\gamma^2)$) to implement the parameter update and t_j is an unknown time delay. Our goal is to obtain for every node j the values of a_j and b_j by measuring only $v_j(t)$, without any knowledge of t_j.

We now establish a necessary and sufficient condition on the nudging signal $f(t)$ so that one can, at least in principle, uniquely obtain the values of a_j and b_j for every node j. We are concerned with quantities that depend

only on a single node and hence will drop the node index with the understanding that all of the analysis applies to any arbitrary node.

Let F denote the set of all real-valued, bounded functions, and let f be an element of F. Let $\mathcal{C}_f : \mathbb{R}^3 \to F$ be the function that maps the parameters (a, b, t_0) to the function $v(\cdot) = a + bf(\cdot + t_0)$. We define the following equivalence relation on F: two functions $g, h \in F$ are equivalent if they differ by a time translation, that is, $g \sim h$, if and only if there exists a $t_0 \in \mathbb{R}$ such that $g(t) = h(t + t_0)$ for all $t \in \mathbb{R}$. Let $\tilde{F} = F/\sim$ be the quotient of F under this equivalence relation and let $[g]$ be the equivalence class that contains the function g. The map $\mathcal{C}_f$ can be lifted to yield $\tilde{\mathcal{C}}_f : \mathbb{R}^2 \to \tilde{F}$ such that $\tilde{\mathcal{C}}_f(a, b) = [a + bf]$. In order to be able to uniquely extract a and b from any equivalence class of the form $[a + bf]$, the function $\tilde{\mathcal{C}}_f$ has to be injective. This can be reexpressed as a direct condition on the nudging signal f.

Proposition 1. *The following statements are equivalent:*

P1: The function $\tilde{\mathcal{C}}_f : \mathbb{R}^2 \to \tilde{F}$ defined by $\tilde{\mathcal{C}}_f(a, b) = [a + bf]$ is injective.
P2: There exists no $\tau \in \mathbb{R}$ such that for all $t \in \mathbb{R}$,

$$f(t) = \sup f + \inf f - f(t + \tau),$$

where $\sup f = \sup_t f(t)$ and $\inf f = \inf_t f(t)$ denote the supremum and infimum values of the nudging signal f, respectively.

Proof. We establish this by proving that the negations of the two statements are equivalent, that is, the following statements are equivalent:

N1: There exist two distinct pairs of real numbers (a_1, b_1) and (a_2, b_2) such that

$$[a_1 + b_1 f] = [a_2 + b_2 f].$$

N2: There exists a $\tau \in \mathbb{R}$ such that for all $t \in \mathbb{R}$,

$$f(t) = \sup f + \inf f - f(t + \tau).$$

Suppose that N2 is true: there is a $\tau \in \mathbb{R}$ such that for all $t \in \mathbb{R}$, $f(t) = \sup f + \inf f - f(t + \tau)$. This means that f and $\sup f + \inf f - f$ are related by a time translation, that is, $[f] = [\sup f + \inf f - f]$. Therefore, N1 is true, with $(a_1, b_1) = (0, 1)$ and $(a_2, b_2) = (\sup f + \inf f, -1)$.

Conversely, suppose that N1 is true: there exist two distinct pairs of real numbers (a_1, b_1) and (a_2, b_2) and a $\tau \in \mathbb{R}$ such that

$$\forall t \in \mathbb{R}, \quad a_1 + b_1 f(t) = a_2 + b_2 f(t + \tau). \tag{B.1}$$

The numbers b_1 and b_2 cannot be both zero; otherwise, equation B.1 implies that $a_1 = a_2$, a contradiction. If $b_1 = 0$ and $b_2 \neq 0$, that equation implies that f is a constant, in which case N2 is clearly true. Otherwise $b_1 \neq 0$, and we can rewrite the above equality as

$$\forall t \in \mathbb{R}, \quad f(t) = a + bf(t + \tau) \tag{B.2}$$

with $a = (a_2 - a_1)/b_1$ and $b = b_2/b_1$. Now there are two possibilities: either $b > 0$ or $b < 0$.

First, let us suppose that $b > 0$. The above equality imposes the following conditions on the minimum and maximum values of the function f:

$$\sup f = a + b \sup f, \tag{B.3}$$

$$\inf f = a + b \inf f. \tag{B.4}$$

Subtracting equation B.4 from B.3 and reorganizing the terms, we get $(1 - b)(\sup f - \inf f) = 0$. If $b = 1$, then $a = 0$, contradicting our assumption that (a_1, b_1) and (a_2, b_2) are distinct pairs. Therefore, $\sup f = \inf f$, f is constant, and N2 is true.

Second, let us suppose that $b < 0$. As before, we have

$$\sup f = a + b \inf f, \tag{B.5}$$

$$\inf f = a + b \sup f, \tag{B.6}$$

and again $(1 + b)(\sup f - \inf f) = 0$. Either f is a constant, or $b = -1$, implying in turn that $a = \sup f + \inf f$. Therefore, coming back to equation B.2, we have $f(t) = \sup f + \inf f - f(t + \tau)$ for all $t \in \mathbb{R}$, which is the statement of N2. □

Appendix C: Multimechanism Learning via Chemical Signaling

In this appendix, we generalize the learning algorithm via chemical signaling (Anisetti, Scellier et al., 2023) to nonlinear networks. Learning via chemical signaling is another example of multimechanism learning in physical networks. It uses pressures and chemical concentrations to implement a local learning rule. This way of using multiple independent "mechanisms" is the central idea behind multimechanism learning.

Consider a flow network, one of nodes interconnected by tubes. A flow network is formally equivalent to the resistive network of section 2, with v being the configuration of node pressures and θ_{jk} being the conductance of the branch between nodes j and k.

Learning via chemical signaling proceeds as follows. In the first phase, given θ and input signals x, the configuration of node pressures stabilizes to its equilibrium value $v(\theta, x)$ given by

$$v(\theta, x) = \arg\min_{v} \; E(\theta, x, v). \tag{C.1}$$

In the second phase, we inject chemical currents $e = -\beta \frac{\partial C}{\partial v}(v(\theta, x), y)$ at output nodes, where β is a (positive or negative) constant. As a result, a chemical concentration u develops at each node. Assuming that the configuration of node pressures $v(\theta, x)$ is not affected by the chemical, the chemical concentration u at equilibrium satisfies the relationship

$$\frac{\partial^2 E}{\partial v^2}(\theta, x, v(\theta, x)) \cdot u = -\beta \frac{\partial C}{\partial v}(v(\theta, x), y). \tag{C.2}$$

Indeed, diffusion along a tube follows the same equation as that of flow along the same tube, up to a constant factor (replacing node pressures and flow conductivity by chemical concentration and diffusion constant, respectively). When there is no ambiguity from the context, we write $v = v(\theta, x)$ for simplicity. We note that although v is not affected by the chemical, u depends on v. In particular u also depends on θ and x through v.

Next, denoting $u = (u_1, u_2, \ldots, u_N)$, we update each parameter θ_{jk} according to the learning rule

$$\Delta\theta_{jk} = -\alpha(u_j - u_k) \cdot (v_j - v_k), \tag{C.3}$$

where α is some constant. Note that this learning rule is local (just like the learning rule of Freq-prop), requiring only information about nodes j and k for each conductance θ_{jk}.

Theorem 3. *For every parameter θ_{jk}, it holds that*

$$\Delta\theta_{jk} = -\alpha\, \beta \frac{\partial \mathcal{L}}{\partial \theta_{jk}}(\theta, x, y). \tag{C.4}$$

Namely, the learning rule of equation C.3 performs one step of gradient descent with respect to the loss, with step size $\alpha\beta$. We note that learning via chemical signaling comes in two variants, either with $\beta > 0$ and $\alpha > 0$ or with $\beta < 0$ and $\alpha < 0$. The procedure performs one step of gradient descent as long as the product $\alpha\beta$ is positive.

Proof. First, we write the first-order equilibrium condition for $v(\theta, x)$, which is

$$\frac{\partial E}{\partial v}(\theta, x, v(\theta, x)) = 0. \tag{C.5}$$

We differentiate this equation with respect to θ:

$$\frac{\partial^2 E}{\partial v^2}(\theta, x, v(\theta, x)) \frac{\partial v}{\partial \theta}(\theta, x) + \frac{\partial^2 E}{\partial v \partial \theta}(\theta, x, v(\theta, x)) = 0. \tag{C.6}$$

Multiplying both sides on the left by $u^\top$, we get

$$u^\top \frac{\partial^2 E}{\partial v^2}(\theta, x, v(\theta, x))\frac{\partial v}{\partial \theta}(\theta, x) + u^\top \frac{\partial^2 E}{\partial v \partial \theta}(\theta, x, v(\theta, x)) = 0. \tag{C.7}$$

On the other hand, multiplying both sides of equation C.2 on the left by $\frac{\partial v}{\partial \theta}(\theta, x)^\top$, we get

$$\begin{aligned}\frac{\partial v}{\partial \theta}(\theta, x)^\top \frac{\partial^2 E}{\partial v^2}(\theta, x, v(\theta, x))u &= -\beta \frac{\partial v}{\partial \theta}(\theta, x)^\top \frac{\partial C}{\partial v}(v(\theta, x), y) \\ &= -\beta \frac{\partial \mathcal{L}}{\partial \theta}(\theta, x, y).\end{aligned} \tag{C.8}$$

Comparing equations C.7 and C.8, we conclude that

$$u^\top \frac{\partial^2 E}{\partial v \partial \theta}(\theta, x, v(\theta, x)) = \beta \frac{\partial \mathcal{L}}{\partial \theta}(\theta, x, y). \tag{C.9}$$

Finally, using the form of the energy function 2.2, we have for each parameter θ_{ij},

$$(u_i - u_j) \cdot (v_i - v_j) = \beta \frac{\partial \mathcal{L}}{\partial \theta_{ij}}(\theta, x, y). \tag{C.10}$$

Therefore, the learning rule

$$\Delta\theta_{jk} = -\alpha(u_i - u_j) \cdot (v_i - v_j) \tag{C.11}$$

satisfies

$$\Delta\theta_{jk} = -\alpha\, \beta \frac{\partial \mathcal{L}}{\partial \theta_{jk}}(\theta, x, y). \tag{C.12}$$

Hence, the result. □

Acknowledgments

We thank Sam Dillavou, Nachi Stern, Andrea Liu, Doug Durian, and Jack Kendall for discussion. J.M.S acknowledges financial support from NSF-DMR-1832002 and NSF-DMR-2204312.

References

Akam, T., & Kullmann, D. M. (2014). Oscillatory multiplexing of population codes for selective communication in the mammalian brain. *Nature Reviews Neuroscience*, *15*(2), 111–122. 10.1038/nrn3668

Anisetti, V. R., Kandala, A., & Schwarz, J. M. (2023). *Emergent learning in physical systems as feedback-based aging in a glassy landscape.* arXiv:2309.04382.

Anisetti, V. R., Scellier, B., & Schwarz, J. M. (2023). Learning by non- interfering feedback chemical signaling in physical networks. *Physical Review Research*, *5*, 023024. 10.1103/PhysRevResearch.5.023024

Baldi, P., & Pineda, F. (1991). Contrastive learning and neural oscillations. *Neural Computation*, *3*(4), 526–545. 10.1162/neco.1991.3.4.526

Bastos, A. M., Vezoli, J., Bosman, C. A., Schoffelen, J.-M., Oostenveld, R., Dowdall, J. R., . . . Fries, P. (2015). Visual areas exert feedforward and feedback influences through distinct frequency channels. *Neuron*, *85*(2), 390–401. 10.1016/j.neuron.2014.12.018

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT 2010* (pp. 177–186).

Boussard, A., Fessel, A., Oettmeier, C., Briard, L., Döbereiner, H.-G., & Dussutour, A. (2021). Adaptive behaviour and learning in slime moulds: The role of oscillations. *Philosophical Transactions of the Royal Society B*, *376*(1820), 20190757. 10.1098/rstb.2019.0757

Cherry, C. (1951). CXVII. Some general theorems for non-linear systems possessing reactance. *London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *42*(333), 1161–1177. 10.1080/14786445108561362

Dillavou, S., Stern, M., Liu, A. J., & Durian, D. J. (2021). *Demonstration of decentralized, physics-driven learning*. arXiv:2108.00275.

Fisher, R. (1988). *Iris.* UCI Machine Learning Repository.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Hopfield, J. J. (1984). 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. 10.1073/pnas.81.10.3088

Kendall, J., Pantone, R., Manickavasagam, K., Bengio, Y., & Scellier, B. (2020). *Training end-to-end analog neural networks with equilibrium propagation*. arXiv:2006.01981.

Laborieux, A., & Zenke, F. (2022). Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in neural information processing systems*, *35* (pp. 12950–12963). Curran.

Lopez-Pastor, V., & Marquardt, F. (2021). *Self-learning machines based on Hamiltonian echo backpropagation.* arXiv:2103.04992.

McCaughan, A. N., Oripov, B. G., Ganesh, N., Nam, S. W., Dienstfrey, A., & Buckley, S. M. (2023). *Multiplexed gradient descent: Fast online training of modern datasets on hardware neural networks without backpropagation.* arXiv:2303.03986.

Millar, W. (1951). CXVI. Some general theorems for non-linear systems possessing resistance. *London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *42*(333), 1150–1160. 10.1080/14786445108561361

Naud, R., & Sprekeler, H. (2018). Sparse bursts optimize information transmission in a multiplexed neural code. *Proceedings of the National Academy of Sciences*, *115*(27), E6329–E6338. 10.1073/pnas.1720995115

Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., & Naud, R. (2021). Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature Neuroscience*, *24*(7), 1010–1019. 10.1038/s41593-021-00857-x

Pereda, A. E. (2014). Electrical synapses and their functional interactions with chemical synapses. *Nature Reviews Neuroscience*, *15*(4), 250–263. 10.1038/nrn3708

Scellier, B. (2021). *A deep learning theory for neural networks grounded in physics*. PhD thesis, Université de Montréal.

Scellier, B., & Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, *11*, 24. 10.3389/fncom.2017.00024

Scellier, B., Mishra, S., Bengio, Y., & Ollivier, Y. (2022). *Agnostic physics-driven deep learning*. arXiv:2205.15021.

Semedo, J. D., Jasper, A. I., Zandvakili, A., Krishna, A., Aschner, A., Machens, C. K., . . . Yu, B. M. (2022). Feedforward and feedback interactions between visual cortical areas use different population activity patterns. *Nature Communications*, *13*(1), 1–14. 10.1038/s41467-022-28552-w

Stern, M., Arinze, C., Perez, L., Palmer, S. E., & Murugan, A. (2020). Supervised learning through physical changes in a mechanical system. *PNAS*, *117*(26), 14842–14850. 10.1073/pnas.2000807117

Stern, M., Hexner, D., Rocks, J. W., & Liu, A. J. (2021). Supervised learning in physical networks: From machine learning to learning machines. *Physical Review X*, *11*(2), 021045. 10.1103/PhysRevX.11.021045

Stern, M., & Murugan, A. (2022). *Learning without neurons in physical systems*. arXiv:2206.05831.

Tweedy, L., Thomason, P. A., Paschke, P. I., Martin, K., Machesky, L. M., Zagnoni, M., & Insall, R. H. (2020). Seeing around corners: Cells solve mazes and respond at a distance using attractant breakdown. *Science*, *369*(6507), eaay9792. 10.1126/science.aay9792

Wright, L. G., Onodera, T., Stein, M. M., Wang, T., Schachter, D. T., Hu, Z., & McMahon, P. L. (2022). Deep physical neural networks trained with backpropagation. *Nature*, *601*(7894), 549–555. 10.1038/s41586-021-04223-6

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

Zucchet, N., & Sacramento, J. (2022). *Beyond backpropagation: Implicit gradients for bilevel optimization*. arXiv:2205.03076.

Received August 15, 2023; accepted November 20, 2023.