Learning by non-interfering feedback chemical signaling in physical networks

Vidyesh Rao Anisetti,[1,*] B. Scellier,[2] and J. M. Schwarz[1,3]
[1]*Physics Department, Syracuse University, Syracuse, New York 13244, USA*
[2]*Department of Mathematics, ETH Zürich, Zürich, Switzerland*
[3]*Indian Creek Farm, Ithaca, New York 14850, USA*

(Received 23 June 2022; accepted 1 April 2023; published 13 April 2023)

Both non-neural and neural biological systems can learn. So rather than focusing on purely brain-like learning, efforts are underway to study learning in physical systems. Such efforts include equilibrium propagation (EP) and coupled learning (CL), which require storage of two different states—the free state and the perturbed state—during the learning process to retain information about gradients. Here, we propose a learning algorithm rooted in chemical signaling that does not require storage of two different states. Rather, the output error information is encoded in a chemical signal that diffuses into the network in a similar way as the activation/feedforward signal. The steady-state feedback chemical concentration, along with the activation signal, stores the required gradient information locally. We apply our algorithm using a physical, linear flow network and test it using the Iris data set with 93% accuracy. We also prove that our algorithm performs gradient descent. Finally, in addition to comparing our algorithm directly with EP and CL, we address the biological plausibility of the algorithm.

DOI: 10.1103/PhysRevResearch.5.023024

I. INTRODUCTION

What basic ingredients constitute a biological learning system, such as slime mold or higher-order organisms? Biological learning systems adapt to the external environment by tailoring specific responses for given external conditions. As the system continues to experience external conditions of a similar kind, it develops functionality to respond to the stimulus in such a way to increase its chances of survival. Intriguingly, this functionality is an emergent phenomenon as a result of interactions between the various components [1]. For example, when birds come together in a flock, they increase their chances of survival [2]. This happens not because of a "supervisor" that commands each bird to fly in a particular way, but because birds, such as starlings, interact with a fixed number of neighbors independent of their density to give rise to emergent functionality [3]. Similarly, in the presence of rising waters, fire ants cooperate to form floating rafts consisting of a structural base and freely-moving ants on top of the base with treadmilling between the two roles [4,5]. Local, ant interaction rules, including an effective repulsive force between the freely-moving ants and the water replicate the types of observed shapes of rafts [6]. These special interactions between components, responsible for emergent functionality in nature, have themselves emerged out of the long process of evolution.

Given the intricacies of biological learning systems, neural networks are *in silico* brain-like learning systems, resembling the visual cortex, in particular [7–10], that can recognize patterns and solve problems [11,12]. More specifically, neural networks achieve functionality by modifying weights and biases to minimise a particular cost function. Of the many ways to do so, the algorithm of choice in neural networks with multiple layers (deep learning) is the backpropagation algorithm [13]. Backpropagation updates the network such that its weights (and biases) perform gradient descent in the cost function landscape. The complex nature of the tasks that neural networks are capable of hints at the possibility that biological learning systems also achieve functionality by optimizing cost functions by gradient descent [14]. In other words, the long process of evolution may have optimised the "learning algorithm" in such biological systems to update its components via gradient descent. The success of backpropagation has, therefore, encouraged a search for biologically plausible learning rules analogous to it [14–20]. For completeness, here are properties one should ensure while constructing such a biologically plausible learning system:

(1) local learning algorithms [21],

(2) the implementation of such algorithms is constrained by the laws of physics, and

(3) the algorithms minimize a cost function via gradient descent or stochastic gradient descent.

Indeed, there have been attempts to construct learning algorithms within purely physical systems [22–28]. Here, we will focus on "equilibrium propagation" [25,29] and "coupled learning"[26]. In these approaches, the error information corresponding to each component is encoded in terms of differences of local physical quantities measured between two learning phases. At each step of training, the outputs of the network are nudged towards the target output by applying additional boundary conditions at the output nodes. Next, the

[*]vvaniset@syr.edu

system is allowed to settle to a new steady state called the "nudged state" (or the "clamped state"), which is closer to the desired target than the initial "free state". In the limit where the nudge amplitude goes to zero, the difference of local, physical quantities between these two learning phases encodes the gradient of the cost function [29]. Unlike backpropagation, these algorithms achieve gradient descent without an explicit layer-by-layer transfer of error information. Nevertheless, one caveat of these approaches is the requirement to store the free state. In other words, the learning rule requires information about the free state, which is no longer physically available at the end of the second phase when the parameters are updated. One way around this requirement is to build two copies of the same physical network [30], although biology does not necessarily have such a luxury.

In this paper, we present an algorithm and a learning rule that overcomes the above requirement in equilibrium propagation and coupled learning. Our learning rule computes the gradients using local information for each weight without the need to store the free state. We demonstrate that the functionality of the nudged state can be realized in physical and biophysical, learning systems using chemical signaling. We show that steady-state chemical concentrations can be used in the second phase to encode the required gradient information. Chemical signaling is ubiquitous in biology. For example, consider the structurally simple, yet functionally complex organism, named *Physarum polycephalum*, otherwise known as slime mold. Slime mold is a unicellular, multinucleated organism that is neither a plant nor an animal nor a fungus. This unicellular organism can span up to the meter scale and consists of a network of tubes whose underlying structure is driven by cytoskeletal reorganization [31]. Despite its simplicity, in the sense that it is non-neuronal, this organism is capable of myriad complex tasks—precisely coordinating flows in its body [32], navigating mazes [33], and connecting food sources with optimal paths [34,35]. Work by K. Alim and others showed that much of this complex phenomenon can be explained by a mechanism of signal propagation [36]. Specifically, slime mold uses a chemical signal to send information regarding the location of food sources across its body, which triggers a change in its tubular structure due to a softening agent to optimize the connection between food sources [37].

In light of an example, we construct a physical learning network of tubes/pipes that uses chemical signals to send error information across the system. Our system is a flow network, with activation pressures at nodes $\mathbf{v}$ and pipe conductance described by weights $\mathbf{w}$. The information from the external environment is input into the system by fixing the boundary conditions $\mathbf{I}$ at input nodes. Node activation pressures $\mathbf{v}$ are the nontrainable variables (or "state variables") of the system that are determined by Laplace's equation and input boundary conditions. Our physical system is, therefore, a linear one. The functionality we seek is to obtain desired pressures ("target pressures") at output nodes for a given input $\mathbf{I}$. To achieve this, a feedback chemical is released into the flow network by fixing the chemical currents at output nodes. The value $\boldsymbol{\epsilon}$ of these chemical currents is proportional to the difference between target pressures and output pressures. The chemical concentrations $\mathbf{u}$ at internal nodes are determined by the same Laplace equation, but with feedback boundary conditions $\boldsymbol{\epsilon}$.

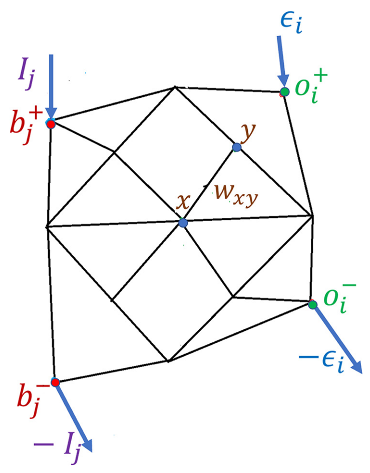

FIG. 1. The flow network. Schematic of the flow network with blue points representing the input node pair and red points representing the output node pair. The weights of the flow network are denoted by w_{xy} between neighboring nodes x and y. These weights are varied during the training-testing process. Input to the network I_j is given as applied boundary currents across pairs of input nodes (b_j^+, b_j^-). Output of the network is measured as pressure drop across output node pairs $v(o_i^+) - v(o_i^-)$ as the response to the applied currents. Feedback is applied by fixing chemical currents ϵ_i across pairs of output nodes (o_i^+, o_i^-).

We show that this chemical concentration $\mathbf{u}$ along with the node pressures $\mathbf{v}$ locally encode the weight gradients of the cost function that we want to optimize. We propose a learning rule that updates the trainable weights $\mathbf{w}$ such that it does gradient descent with respect to the cost function.

II. THEORY

We consider a flow network of nodes interconnected by weighted edges (see Fig. 1). We denote w_{xy} the weight (i.e., conductance) of the edge between node x and node y. A subset of the nodes are boundary node pairs (or "input" node pairs), denoted $\{(b_1^+, b_1^-), (b_2^+, b_2^-), \dots, (b_q^+, b_q^-)\}$. For each pair (b_j^+, b_j^-), an input current I_j flows into the network through the node b_j^+ and flows out of the network through the node b_j^-. The state of the system is defined by the node pressures, denoted $v(x)$ and governed by Laplace's equation at steady state. Another subset of the nodes are output node pairs $\{(o_1^+, o_1^-), (o_2^+, o_2^-), \dots, (o_p^+, o_p^-)\}$. The output of the network is defined as the set of pressure drops across output nodes $\{v(o_1^+, o_1^-), v(o_2^+, o_2^-), \dots, v(o_p^+, o_p^-)\}$ where $v(o_i^+, o_i^-) = v(o_i^+) - v(o_i^-)$. We note that $v(o_i^+, o_i^-)$ is a function of input currents $\{I_j\}$ and all the weights of the network $\{w_{xy}\}$. Training the network consists in modifying the weights $\{w_{xy}\}$ such that, given the input currents $\{I_j\}$, we get desired pressure drops $\{v_d(o_i^+, o_i^-)\}$ across the output node pairs. We define the cost function

$$C = \frac{1}{2} \sum_{i=1}^{p} (v(o_i^+, o_i^-) - v_d(o_i^+, o_i^-))^2. \qquad (1)$$

Now we present a physical procedure and a learning rule for the weights that performs gradient descent with respect to the cost function. To achieve this, we release a feedback chemical into the network through the pairs of output nodes $\{(o_i^+, o_i^-)\}$, by fixing chemical currents across these nodes (see Fig. 1). Specifically, for each pair of output nodes (o_i^+, o_i^-), a current

$$\epsilon_i = \eta(v_d(o_i^+, o_i^-) - v(o_i^+, o_i^-)) \tag{2}$$

flows into the network through node o_i^+ and flows out of the network through node o_i^-, where η is a constant ("nudging"). A steady-state chemical concentration develops at every node, governed again by Laplace's equation. We denote $u(x)$ the steady-state concentration at node x, and $u(x, y)$ the drop in chemical concentration between nodes x and y. We regard quantities v and u as givens for now. The precise equations governing them will be stated later. Finally, we update each weight w_{xy} according to

$$\Delta w_{xy} = -\alpha u(x, y)v(x, y), \tag{3}$$

where α is a constant. We show below that this learning rule performs gradient descent on the cost function with step size ("learning rate") $\alpha\eta$, i.e.,

$$\Delta w_{xy} = -\alpha\eta\frac{\partial C}{\partial w_{xy}} \tag{4}$$

for every weight w_{xy}. The above learning rule for the weights is local. The final error term depends upon two quantities; the pressure drop due to flow and the concentration drop of the feedback chemical. If they have the same sign, then the weight gets a positive update and vice versa. Here, we assume that the relaxation time scale of the system is much faster than the time scale of weight updates so that the system is in steady state as the weights are updated.

Note that the two quantities in the weight update are independent of each other, therefore, we assume that diffusion is fast enough that it is independent of the flow. In an experimental setting, one can realize this by using very small flow rate, leading to very small pressure drops across weights and using a signaling chemical with very high diffusion rate via, perhaps, some catalytic process. We understand that such a construction is not necessarily realized in nature; therefore, we also propose a purely flow version of the model where the chemical signals are carried by the fluid flow and not by diffusion (see Appendix A). While this chemical flow algorithm is presumably more physically plausible, it is not yet clear that the algorithm performs gradient descent. In any event, what we present here is an idealization. Obviously, nature may be using a complex combination of flow and diffusion for signaling.

Considering v as a pressure and u as a chemical concentration is just a certain packaging of the theory. The central idea of this paper is to use two independent physical quantities, which leads to non-interference of the input signal and the feedback signal, in other words one can use any two non-interfering modalities to conduct learning [38]. For example, one can use two chemicals v and u diffusing in a static fluid, with distinct chemical signatures to encode input and error signal. In this case, there is no need for an additional assumption on the relationship between flow rate and diffusion rate.

Our result holds for any cost function C, not just the squared error [Eq. (1)]. In general, in the second phase, the chemical current flowing in through o_i^+ and flowing out through o_i^- must be $\epsilon_i = -\eta\frac{\partial C}{\partial v(o_i^+, o_i^-)}$. Our result also holds if we reverse the sign of η (2) and the sign of α in the learning rule (3). In other words, the algorithm performs gradient descent so long as $\alpha\eta > 0$.

Now we prove our claim that the learning rule (3) performs gradient descent (4). Let us number the nodes of the network $1, 2, \dots, n$. Let I_x be the input current at node x (with $I_x = 0$ by convention if node x is not an input node) and v_x the pressure at node x. For each node x, the steady-state condition at node x in the first phase yields

$$\sum_y w_{xy}(v_x - v_y) = I_x, \tag{5}$$

which is the current conservation equation. The summation here and subsequent ones are taken over all y that are neighbors of x, and there are no self connections ($w_{xx} = 0$). We, thus, arrive at a system of n linear equations. This system rewrites with matrix-vector notations as

$$L \cdot v = I, \tag{6}$$

where v is the vector of node pressures, I is the vector of input currents, and L is the matrix

$$L = \begin{pmatrix} \sum_x w_{1x} & -w_{12} & -w_{13} & \cdots & -w_{1n} \\ -w_{21} & \sum_x w_{2x} & -w_{23} & \cdots & -w_{2n} \\ -w_{31} & -w_{32} & \sum_x w_{3x} & \cdots & -w_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -w_{n1} & -w_{n2} & -w_{n3} & \cdots & \sum_x w_{nx} \end{pmatrix}. \tag{7}$$

Note that the matrix L is symmetric because $w_{xy} = w_{yx}$ for every pair of nodes (x, y). Next, we denote $E_x = -\eta\frac{\partial C}{\partial v_x}$ the chemical current at node x in the second phase (with $E_x = 0$ by convention if node x is not an output node), and u_x the concentration of the chemical at node x. Assuming that the diffusion constant of the chemical is equal to the flow conductivity (up to a constant of proportionality), the chemical concentration at steady state satisfies the same system of linear equations, with different boundary conditions, i.e.,

$$L \cdot u = E, \tag{8}$$

where u is the vector of node concentrations, and E is the vector of chemical currents. Now we compute the gradient of the cost function: for every weight w_{xy} we have

$$\frac{\partial C}{\partial w_{xy}} = \left(\frac{\partial C}{\partial v}\right)^\top \cdot \frac{\partial v}{\partial w_{xy}}. \tag{9}$$

Multiplying by $-\eta$ on both sides, we get, by definition of E,

$$-\eta\frac{\partial C}{\partial w_{xy}} = E^\top \cdot \frac{\partial v}{\partial w_{xy}}. \tag{10}$$

Using the steady-state condition of the second phase (8) and the fact that L is symmetric, we get

$$-\eta\frac{\partial C}{\partial w_{xy}} = u^\top \cdot L \cdot \frac{\partial v}{\partial w_{xy}}. \tag{11}$$

Next, we differentiate the steady-state condition of the first phase (6) with respect to w_{xy}. We get

$$\frac{\partial L}{\partial w_{xy}} \cdot v + L \cdot \frac{\partial v}{\partial w_{xy}} = 0. \tag{12}$$

Rearranging the terms and injecting this in (11), we get

$$\eta \frac{\partial C}{\partial w_{xy}} = u^\top \cdot \frac{\partial L}{\partial w_{xy}} \cdot v. \tag{13}$$

Looking at the form of the matrix L (7), the matrix $\frac{\partial L}{\partial w_{xy}}$ has exactly four nonzero coefficients: those at positions (x, x), (x, y), (y, x), and (y, y), equal to $+1$, -1, -1, and $+1$, respectively. Therefore

$$u^\top \cdot \frac{\partial L}{\partial w_{xy}} \cdot v = u(x)v(x) - u(x)v(y) - u(y)v(x) + u(y)v(y) \tag{14}$$

$$= (u(x) - u(y)) \cdot (v(x) - v(y)) \tag{15}$$

$$= u(x, y) \cdot v(x, y). \tag{16}$$

Combining this with (13), we conclude that

$$\Delta w_{xy} = -\alpha u(x, y) \cdot v(x, y) = -\alpha\eta \frac{\partial C}{\partial w_{xy}}. \tag{17}$$

Hence, the learning rule corresponds to one step of gradient descent with step size $\alpha\eta$. This concludes the proof.

Summing up the training mechanism:

(1) A flow network is generated where chemicals can spread via a diffusion process.

(2) Input to the network is given by fixing currents $\{I_j\}$ across input node pairs $\{(b_j^+, b_j^-)\}$, leading to steady-state pressures v. Outputs are measured as pressure drops $\{v(o_i^+, o_i^-)\}$ across output node pairs $\{(o_i^+, o_i^-)\}$.

(3) Outputs are compared to the desired outputs $\{v_d(o_i^+, o_i^-)\}$, and a feedback chemical is released in the network by fixing the chemical currents $\epsilon_i = \eta(v_d(o_i^+, o_i^-) - v(o_i^+, o_i^-))$ across the output node pairs $\{(o_i^+, o_i^-)\}$. The chemical concentration reaches steady state u.

(4) The concentration drop $u(x, y)$ and pressure drop $v(x, y)$ are measured across each weight w_{xy}, and the weights are updated according to $\Delta w_{xy} = -\alpha u(x, y)v(x, y)$.

(5) This procedure, which corresponds to one step of gradient descent, is repeated iteratively until convergence of the weights is achieved.

III. THE IRIS DATA SET

We train the flow network on a standard machine learning task: classifying Iris flowers. The Iris data set [39] 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 cm), 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.

We now detail how we do this. A flow network is constructed as follows:

(1) Generate a square lattice,

(2) Perturb the positions of the lattice with a step of length δ in any random direction,

(3) Every node is connected to its d nearest neighbors [40]. We choose $d = 4$ for all our simulations unless specified.

(4) Every edge of the network is assigned a conductance from a truncated normal distribution.

(5) Four pairs of input nodes are chosen from this flow network, where the input data (the normalized features of the Iris) is given as external currents across these four pairs of boundary nodes. Three pairs of nodes are chosen as the output nodes such that every pair is composed of two neighboring nodes. For pairs of output nodes that are not neighboring nodes, the training error decreased less smoothly. Once the network is trained, for a given input, the set of potential drops across these node pairs should tell the category of Iris the input data corresponds 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 have 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 $X_i^{\text{norm}} = \lambda \cdot \frac{X_i - X_i^{\text{min}}}{X_i^{\text{max}} - X_i^{\text{min}}}$, 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 (see Appendix B). In other words, we must choose desired outputs that are physically attainable. We, therefore, implement the same technique as described in Ref. [30] 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, 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.

We also implement the conventional one-hot encoding technique commonly utilized in machine learning (refer to Appendix B). However, due to physical constraints, we observe that this set of desired outputs exhibits subpar performance in comparison to the aforementioned method.

To conduct the training process, first the input data is given to the network and the output is observed. If the output is not equal to the desired output for that Iris category, feedback chemical is released at the output node pairs. This is done by applying a constant chemical current. The weights of the network are modified using the learning rule mentioned above. This process is repeated consecutively for all examples. Next, once the entire data set is exhausted, we say "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.

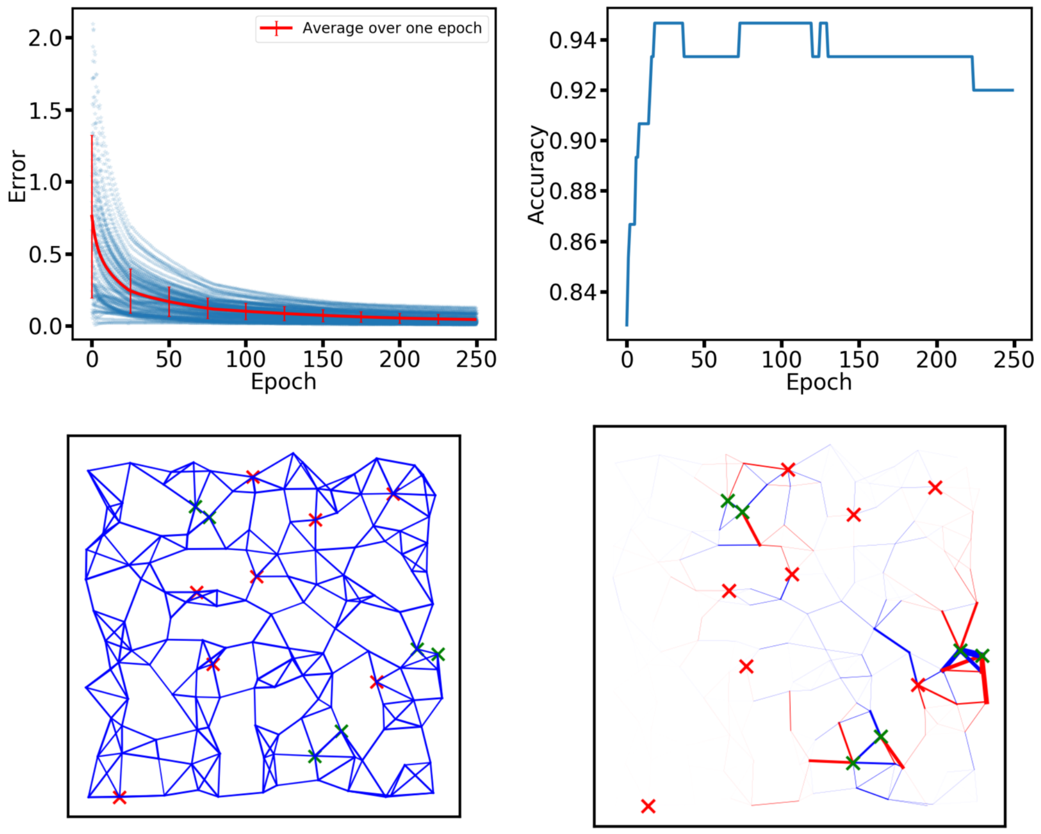

FIG. 2. Training and testing using the Iris data set. The Iris data set is trained on a network with 12^2 nodes and learning rate $\eta = 10^{-4}$. The weights are sampled from a truncated normal distribution with mean of 0.1, standard deviation of 0.01 with a lower cutoff of 0.01 and upper cutoff of 0.19. We use the standard mean squared error (L2 norm) as the cost function. (Top left) At each training step, the network is shown an example and the mean squared error is calculated between the network's output and desired output, shown as a blue dot. (Top right) Accuracy is defined as the fraction of correct predictions out of total testing examples. (Bottom left) The initial state of the network, with thickness of blue edges representing the conductance. Red crosses denote input nodes; green crosses denote output nodes. (Bottom right) The change in weights between the initial state and the final trained state of the network. Red denotes negative change, while blue denotes positive change with thickness showing the magnitude of change.

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.

The aforementioned training-testing procedure is implemented on a biophysical network. Figure 2 shows the training error and testing accuracy for training using the Iris data set. This training procedure is also implemented on the Wine data set [41] (refer to Appendix C).

IV. LINK WITH EQUILIBRIUM PROPAGATION AND COUPLED LEARNING

To readily see the link with our algorithm and EP, suppose that, in the second phase, instead of injecting the chemical, we inject more of the main substance at output nodes [through the currents $\epsilon_i = \eta(v_d(o_i^+, o_i^-) - v(o_i^+, o_i^-))$]. We can adapt the analysis of Sec. II to show that, instead of the chemical concentration, u now represents the pressure difference between the second phase and the first phase (after and before injecting the currents ϵ_i at output nodes). Indeed, denoting v^{free} and v^{nudged} the node pressures at equilibrium in the first phase and second phase, respectively, we have

$$L \cdot v^{\mathrm{free}} = I, \tag{18}$$

$$L \cdot v^{\mathrm{nudged}} = I - E, \tag{19}$$

where we recall that E is the vector of currents at output nodes. Equation (18) is the same Laplace equation as (6), therefore $v^{\mathrm{free}} = v$. Moreover, subtracting (19) from (18), we see that $v^{\mathrm{nudged}} - v^{\mathrm{free}}$ satisfies the same equation (8) as the chemical concentration u, that is $L \cdot (v^{\mathrm{nudged}} - v^{\mathrm{free}}) = -E$. We conclude that $u = v^{\mathrm{nudged}} - v^{\mathrm{free}}$. We thus recover the setting of equilibrium propagation [24] with "free state" $v^{\mathrm{free}} = v$ and "nudged state" $v^{\mathrm{nudged}} = v + u$. Coupled learning [26] is also very closely related, with the difference that nudged states are realized by imposing boundary conditions on the pressures of output nodes, rather than on the currents.

Without loss of generality, let us assume for convenience that $\alpha = -1$. Our learning rule rewritten in terms of the free and nudged states is

$$\Delta w_{xy} = -v(x,y)u(x,y) \tag{20}$$

$$= \tfrac{1}{2}v(x,y)^2 + \tfrac{1}{2}u(x,y)^2 - \tfrac{1}{2}(v(x,y)+u(x,y))^2 \tag{21}$$

$$= \tfrac{1}{2}(v^{\text{free}}(x,y)^2 - v^{\text{nudged}}(x,y)^2) + O(\eta^2), \tag{22}$$

where η is the nudge amplitude, i.e., the factor that scales the amplitude of the currents injected in the second phase. Here we have used that u is proportional to the nudge amplitude, i.e., $u(x,y) = O(\eta)$. We have thus recovered the learning rule of equilibrium propagation [24] and coupled learning [26], which, at order 1 in η, is

$$\Delta w_{xy}^{\text{EP/CL}} = k((v^{\text{free}}(x,y))^2 - (v^{\text{nudged}}(x,y))^2). \tag{23}$$

In equilibrium propagation and coupled learning, the multiplicative factor in front of the learning rule is $k = \frac{\text{learning rate}}{2\times\text{nudge amplitude}}$; in our setting here, since $\alpha = -1$, the nudge amplitude is the same as the learning rate, therefore $k = \frac{1}{2}$.

To understand the discrepancy of $O(\eta^2)$ between our learning rule and the learning rule of EP and CL, let us rewrite v^{nudged} as v^η to explicitly show the dependence of the nudged state on the nudge amplitude η. In particular, $v^0 = v^{\text{free}} = v$. In our linear flow network, u is a linear response of η [specifically $u = -\eta L^{-1}\frac{\partial C}{\partial v}$ [42], see Eq. (8)]. Combined with the relationship $u = v^\eta - v^0$ shown above, this implies that $u = \eta \frac{\partial v^\eta}{\partial \eta}|_{\eta=0}$. Therefore

$$\Delta w_{xy} = -v(x,y)u(x,y) \tag{24}$$

$$= -\eta\, v(x,y)\frac{\partial v^\eta(x,y)}{\partial \eta}\bigg|_{\eta=0} \tag{25}$$

$$= -\frac{\eta}{2}\frac{d}{d\eta}\bigg|_{\eta=0}[v^\eta(x,y)]^2. \tag{26}$$

We recover the learning rule $\Delta w_{xy}^{\text{EP/CL}}$ of Eq. (23) as a finite difference approximation of Eq. (26)—hence the term $O(\eta^2)$.

The advantage of our method over EP and CL is that, by using chemical signaling, the system components can now differentiate between "activation" and "feedback" signals based on their distinct chemical signatures. This happens because the chemical u encodes the same information as $\frac{\partial v^\eta}{\partial \eta}$. Not only this eliminates the need to store information about two separate learning phases, but also gives a way for exact gradient computation.

On the other hand, one of the strengths of equilibrium propagation and coupled learning is that they can be used for arbitrary physical systems driven by physical equilibration: their learning rule can be written in terms of derivatives of energy with respect to the trainable parameters [25,26,29], as follows:

$$\Delta \mathbf{w}^{\text{EP/CL}} = 2k\left(\frac{\partial \mathcal{E}^{\text{free}}}{\partial \mathbf{w}} - \frac{\partial \mathcal{E}^{\text{nudged}}}{\partial \mathbf{w}}\right), \tag{27}$$

where $\mathcal{E}^{\text{free}}$ is the energy of the system in the free phase and $\mathcal{E}^{\text{nudged}}$ is the energy in the nudged phase. For example, Eq. (23) can be obtained from Eq. (27) by taking $\mathcal{E}$ as power dissipation function [24,26]. While the present paper focuses on *linear* physical systems, we will leave the study of our algorithm in nonlinear systems for future work.

V. DISCUSSION

We present a simple model of a physical learning system that learns via chemical signaling. In our system, the error between the desired behavior and observed behavior at the output nodes is encoded in the form of a feedback chemical signal. These signals travel across the network via diffusion with the weights of the network updating in response to the concentration of the feedback chemical and there is no need for two states, as there is in equilibrium propagation and coupled learning. We also show that this learning rule minimizes a cost function via gradient descent even beyond the infinitesimal nudge amplitude or learning rate limit. Our simple model allows a physical system to learn complex tasks. However, is not yet optimized for computational efficiency to able to compete with current benchmark results, as evidenced in Ref. [39]. While computational efficiency is not yet a current goal of our paper, such efficiency may be feasible in a nonlinear adaption in the near future.

Given the prevalence of backpropagation, we also compare our algorithm to backpropagation. While there are many explicit differences between backpropagation and our algorithm—mostly because artificial neural networks are very different from physical flow networks—we can compare the basic ideas between these algorithms. In our model the weight update rule is proportional to two quantities: the "error term" $u(x,y)$, which tells how much the pressure drop across the weight w_{xy} must change, and the "activation term" $v(x,y)$, which tells the existing pressure drop due to input. This is similar to backpropagation where the weight update is proportional to the presynaptic input and error in the postsynaptic output [11]. This "activation" times "error" term in the learning rule is reminiscent of Hebbian learning [43]. However, the difference between these algorithms is seen in the way error information is communicated. In backpropagation, the error at the output layer and the weights projecting onto this output layer determines the error at the penultimate layer and so on [11]. Therefore, the relationship between the error values of two layers only depends on the local weight values connecting them. In our model, once the error at output is known, the error at the neighboring nodes is determined by the steady state of the system, which means that the relationship between their error depends on all the weights of the network.

While our paper here is mostly concerned with *linear* flow networks in the absence of advection (see Appendix A for an exception to this), one can explore how our algorithm can be extended to include it. Moreover, our physical procedure (Sec. II) may also be applied to nonlinear systems, i.e., systems whose components have nonlinear characteristics. We will leave the mathematical analysis and experimentation of these settings for future work. We also must explore going beyond the quasistatic limit as time scales also constrain biological learning systems. Efforts towards this goal have recently been made *in silico* and in experiments using coupled learning [44]. Similar extensions can be implemented using our algorithm.

Nature may indeed be using similar signaling mechanisms that we have elaborated on here. Cells use biochemical signals to structure themselves in response to external conditions to optimize their functionality and, thus, identity [45]. Slime mold, a tubular network-like single cell organism, uses chemical signals to modify its tube radii in response to food as the external stimuli [36]. While the particular chemical still remains unknown, a recently proposed candidate is ATP [37]. ATP is crucial for the activity of myosin and, hence, the contractility of the actin cytoskeleton. An increase in ATP would presumably enhance contractility. One might conclude that enhanced contractility leads to stiffer cells as is found in cells adherent to a substrate. However, for cells in suspension, inhibition of myosin leads to stiffer cells due to accelerated depolymerization of actin filaments [46]. Perhaps suspended cell behavior is more relevant for the tubular structures in slime model than adherent cells behavior. On the other hand, given such understanding, we expect similar artificial versions of this signaling mechanism could be used for experimental realizations of our model with its aforementioned extensions. For instance, an experimental system that makes use of different mechanisms, such as different catalytic reactions on a flexible sheet to drive shape changes in the presence of flows (see Appendix A), is a distinct possibility [47]. Alternatively, perhaps one may implement similar catalytic reactions in stiff sheets that fold. Incidentally, supervised learning in stiff sheets, i.e., origami, has been explored [48].

As for multicellular organisms, the brain presumably fine tunes the synaptic strengths of billions of neurons to generate an optimal behavior. For this to happen, feedback signals should not only carry precise credit information to individual neurons but while doing so, they must not interfere with the activation/feedforward signals [14]. How the brain does this is still unknown and many models have been developed to explain this phenomenon. For instance, some models invoke the use of error neurons [49], while others assume a temporal segregation of activation and feedback learning phases [26]. Others propose a compartmentalization of individual neurons to spatially separate information as opposed to temporal segregation [50]. Our learning mechanism is similar to this last one. It avoids multiple learning phases by modifying a node to store two kinds of information—an activation signal and a feedback signal. The reason they do not interfere is because the system components can identify them by their chemical signatures—a ubiquitous phenomenon in nature. We have shown how the same network structure used to send an activation signal, can be used to send precise gradient information to individual weights. Therefore, our model optimizes a cost function via gradient descent, something that deep neural networks already do to achieve human like functionality [51–54]. In light of the reasons stated above, we think our model may ultimately help neuroscientists understand credit assignment mechanisms in the brain.

ACKNOWLEDGMENTS

The authors thank Sid Mishra, Sam Dillavou, Doug Durian, Andrea Liu, and Nachi Stern for discussion. J.M.S. acknowledges funding support from Grant No. NSF-DMR-1832002 and an Isaac Newton Award from the DoD.

APPENDIX A: FLOW VERSION OF THE CHEMICAL SIGNALING ALGORITHM

Here we present a model where the chemical signal spreads not via diffusion but via advection. To begin, the pressure at a node depends on the resistance to flow downstream. Therefore, to alter the pressure at a node, we must change the conductance of pipes downstream. Let us say that there is an output node pressure we wish to decrease. We will simply release a chemical at that node, which gets carried by the current downstream. This chemical is such that, when it is flowing through a pipe, it increases the conductance of the pipe (e.g., making it thicker). This increase in conductance decreases the resistance to flow, which in turn decreases the pressure at the output node. Similarly, when we wish to increase the output node pressure, we must release a different kind of chemical, which decreases the conductance of the pipes (e.g., making it thinner). Using this we can tune the network to output desired voltages. In fact, we observe numerically system optimizes a cost function—but not necessarily via gradient descent.

To implement the above idea as a tuning process, consider

(1) The input pressures $\{p_i\}$ is applied at output nodes. A supervisor checks the output pressures $\{v\}$ at output nodes and compares them to desired output pressures $\{v_d\}$.

(2) There are two kinds of chemicals, s_+ and s_-. s_+ increases the conductance of the pipe when it passes through it, and vice versa for s_-. We assume that the output nodes release a chemical whose amount is proportional to the difference between the present output pressures and desired output pressures. At $t = 0$ for some output node a, $v(a) \neq v_d(a)$, then

$$if\, v(a) > v_d(a) \Rightarrow s_+(a) = \lambda(v(a) - v_d(a)) \quad \text{at} \quad t = 0, \tag{A1}$$

$$elif\, v(a) < v_d(a) \Rightarrow s_-(a) = \lambda(v_d(a) - v(a)) \quad \text{at} \quad t = 0, \tag{A2}$$

Where λ is the factor that controls the chemical response given by the node to the difference in pressures. Moreover, $s_+(a)$ denotes amount of chemical (e.g., no of molecules) at node“a”.

(3) This chemical is carried by the current in the network. Therefore, in the next time step the chemical flows to the neighboring nodes of a that are downstream to a [55]. We call all such downstream neighbors of a as $\mathcal{D}(a)$. Then for all $b \in \mathcal{D}(a)$,

$$s_+(b, t+1) = s_+(a, t) \times \frac{i(b, a)}{\sum_{x\in\mathcal{D}(a)} i(x, a)} + \text{(incoming chemical from other nodes)}, \tag{A3}$$

where $i(x, a)$ represents the current from a to x. Note that all the chemical initially present at a flows downstream after one time step.

(4) Using the above equation, an N × N array $\hat{S}_+$ is generated, where each entry i, j denotes the amount of chemical passing through the pipe $\{i, j\}$ at step $t \rightarrow t+1$. Let $\hat{W}$ denote the conductance matrix of the graph, where each entry $\{i, j\}$ denotes the conductance of that pipe. Then

$$\hat{W}(t+1) = \hat{W}(t) + \xi(\hat{S}_+ - \hat{S}_-), \tag{A4}$$

ξ controls the response of the pipe to the passing chemical.

(5) The new potentials are calculated on the interior vertices using $\hat{W}(t+1)$. Again, the supervisor checks if $v(a) = v_d(a)$. The chemical takes some time to reach the boundary nodes, where it drains out of the network [56]. Therefore, the total change in potential due to the chemical released at the output nodes is observed after some amount of time. Therefore, we introduce a time delay τ before releasing the chemical once again [57].

(6) This process is repeated iteratively.

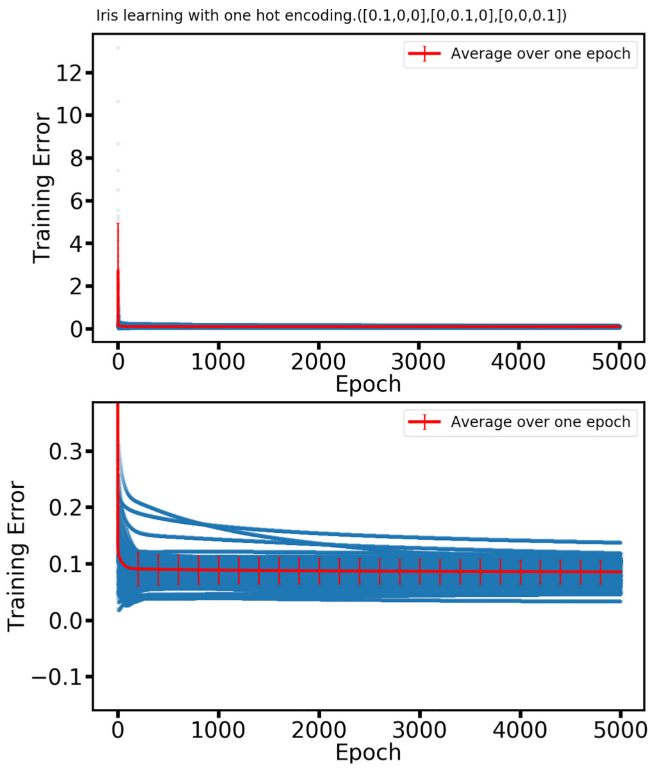

FIG. 3. Training error using the Iris data set with one hot encoding. The Iris data set is trained on a network with 12^2 nodes and learning rate $\eta = 10^{-4}$. The weights are sampled from a truncated normal distribution with mean of 0.1, standard deviation of 0.01 with a lower cutoff of 0.01 and upper cutoff of 0.19. We use the standard mean squared error (L2 norm) as the cost function. The target pressure drops for each Iris category is [0.1, 0, 0], [0, 0.1, 0], [0, 0, 0.1], respectively. (Top) At each training step, the network is shown an example and the mean squared error is calculated between the network's output and desired output, shown as a blue dot. (Bottom) Zoomed version of the above plot. Note that the training error slightly goes below 0.1 (see Fig. 4 for accuracy on testing data).

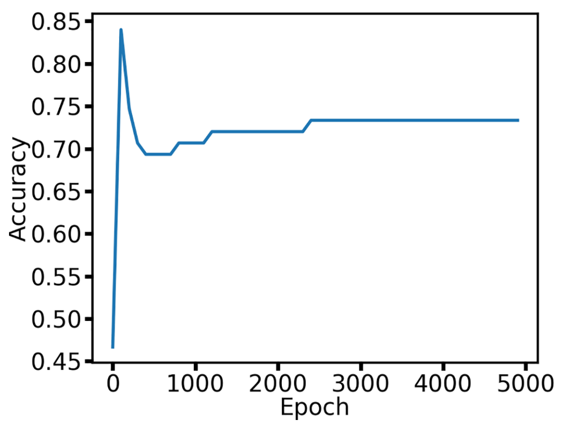

FIG. 4. Testing accuracy using the Iris data set with one hot encoding (for Fig. 3). Accuracy is defined as the fraction of correct predictions out of total testing examples.

APPENDIX B: PHYSICAL CONSTRAINTS AND TRAINING USING ONE HOT ENCODING

Here, we train a linear physical system to approximate a function that maps Iris features to the species it belongs

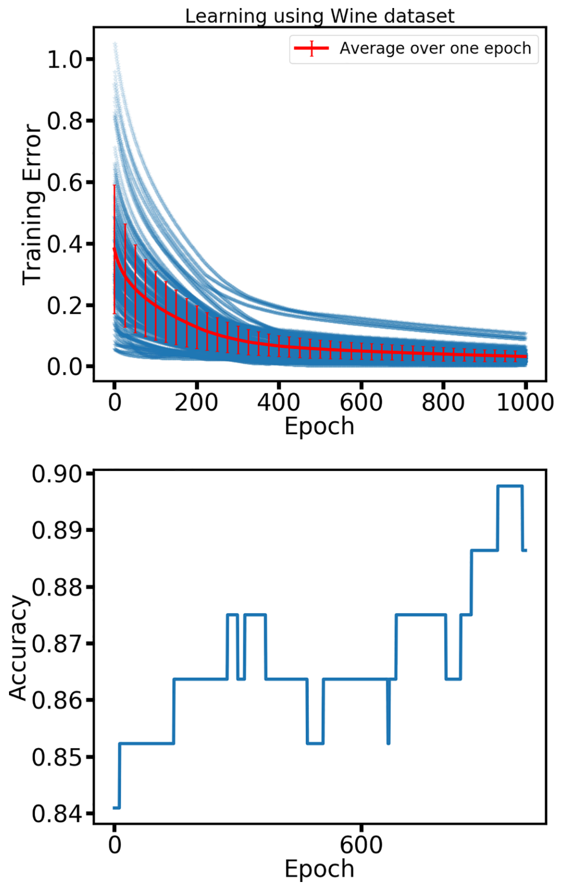

FIG. 5. Training using Wine data set. Trained on a network with 12^2 nodes and learning rate $\eta = 10^{-4}$. The weights are sampled from a truncated normal distribution with mean of 0.1, standard deviation of 0.01 with a lower cutoff of 0.01 and upper cutoff of 0.19. We use the standard mean squared error (L2 norm) as the cost function. (Top) Training error vs epoch. (Bottom) Accuracy vs epoch.

to. Unlike artificial neural networks, these physical systems are restricted by physical constraints, due to which the set of attainable states of the system is limited. Therefore, it is not necessary that any choice of target is physically plausible. For example; if $\{v_d(o_i)\}$ is the set of desired target pressures at the output nodes, and $\{v(b_j)\}$ is the set of input pressure applied at the boundary nodes, then $min\{v(b_j)\} < v_d(o_i) < max\{v(b_j)\}$ for all boundary nodes o_i (and is also known as min-max theorem). Such a constraint arises because Laplace's equation ensures that $\nabla^2 v = 0$ at all interior nodes and only at boundary nodes $\nabla^2 v \neq 0$. In other words, the potential landscape has peaks and minima only at the boundary nodes, and all other interior pressures must lie on the slopes of this landscape. Such a property leads to strong constraints, which is not just limited to min-max theorem stated above. Therefore, the best way to ensure that the desired pressures are physically attainable is to set the target pressure as the class mean output. This method was first incorporated in work done by Ref. [30] to implement coupled learning in physical systems.

We explored the training performance with one hot encoding using the Iris data set (see Figs. 3 and 4). The system still decreases training error and increases accuracy significantly, but because of physical constraints it no longer achieves a high classification accuracy. One can see in Fig. 3 that the system slows down reducing the training error after the 0.1 mark, which corresponds to the one hot encoded target pressure drop. This slowing down means that on an average the output pressure drops are close to [0,0,0], therefore we see a saturating mean squared error close to 0.1. This happens because the system cannot find a set of weights that brings the class output closer to the target pressure corresponding to that class. The one hot encoded targets are not physically favourable and on an average the system's output remains close to [0,0,0].

APPENDIX C: TRAINING USING WINE DATASET

The network is now also trained on Wine dataset [41] (see Fig. 5). The Wine dataset has 13 attributes and three output classes. With 178 total data points, 90 were used for training and rest for testing. Due to the large number of attributes, the network does not perform as well as the Iris dataset.

[1] D. J. Sumpter, The principles of collective animal behaviour, Philos. Trans. R. Soc. B **361**, 5 (2006).

[2] G. Beauchamp, Flocking in birds increases annual adult survival in a global analysis, Oecologia **197**, 387 (2021).

[3] M. Ballerini *et al.*, Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study, Proc. Natl. Acad. Sci. USA **105**, 1232 (2008).

[4] N. J. Mlot, C. A. Tovey, and D. L. Hu, Fire ants self-assemble into waterproof rafts to survive floods, Proc. Natl. Acad. Sci. USA **108**, 7669 (2011).

[5] B. J. Adams, L. M. Hooper-Bui, R. M. Strecker, and D. M. O'Brien, Raft formation by the red imported fire ant *Solenopsis invicta*, J. Insect. Sci. **11**, 171 (2011).

[6] R. J. Wagner and F. J. Vernerey, Computational exploration of treadmilling and protrusion growth observed in fire ant rafts, PLoS Comput. Biol. **18**, e1009869 (2022).

[7] E. B. Issa, C. F. Cadieu, and J. J. Dicarlo, Neural dynamics at successive stages of the ventral visual stream are consistent with hierarchical error signals, eLife **7**, e42870 (2018).

[8] N. Kriegeskorte, Deep neural networks: A new framework for modeling biological vision and brain information processing, Annu. Rev. Vision Sci. **1**, 417 (2015).

[9] D. Zipser and R. Andersen, A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons, Nature (London) **331**, 679 (1988).

[10] P. R. Roelfsema and A. Holtmaat, Control of synaptic plasticity in deep cortical networks, Nat. Rev. Neurosci. **19**, 166 (2018).

[11] M. A. Nielsen, "Neural networks and deep learning," (2018).

[12] G. R. Yang and X. J. Wang, Artificial neural networks for neuroscientists: A primer, Neuron **107**, 1048 (2020).

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).

[14] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, Backpropagation and the brain, Nat. Rev. Neurosci. **21**, 335 (2020).

[15] J. Sacramento, R. Ponte Costa, Y. Bengio, and W. Senn, Dendritic cortical microcircuits approximate the backpropagation algorithm, in Advances in Neural Information Processing Systems, Vol. 31 (2018).

[16] I. Pozzi, S. Bohté, and P. Roelfsema, A biologically plausible learning rule for deep learning in the brain, arXiv:1811.01768.

[17] B. A. Richards and T. P. Lillicrap, Dendritic solutions to the credit assignment problem, Curr. Opin. Neurobiol. **54**, 28 (2019).

[18] J. C. Whittington and R. Bogacz, Theories of error backpropagation in the brain, Trends Cognit. Sci. **23**, 235 (2019).

[19] A. Payeur, J. Guerguiev, F. Zenke, B. A. Richards, and R. Naud, Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits, Nat. Neurosci. **24**, 1010 (2021).

[20] S. Kan, K. Nakajima, T. Asai, and M. Akai-Kasaya, Physical implementation of reservoir computing through electrochemical reaction, Adv. Sci. **9**, 2104076 (2021).

[21] By local we not only mean in terms of metric distance.

[22] J. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, Proc. Nat. Acad. Sci. USA **81**, 3088 (1984).

[23] D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, Physics for neuromorphic computing, Nat. Rev. Phys. **2**, 499 (2020).

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

[25] B. Scellier, A deep learning theory for neural networks grounded in physics, PhD. thesis, Université de Montréal, 2021.

[26] M. Stern, D. Hexner, J. W. Rocks, and A. J. Liu, Supervised Learning in Physical Networks: From Machine Learning to Learning Machines, Phys. Rev. X **11**, 021045 (2021).

[27] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, Deep physical neural networks trained with backpropagation, Nature (London) **601**, 549 (2022).

[28] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: A review, Neural Netw. **115**, 100 (2019).

[29] B. Scellier and Y. Bengio, Equilibrium propagation: Bridging the gap between energy-based models and backpropagation, Front. Comput. Neurosci. **11**, 24 (2017).

[30] S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian, Demonstration of Decentralized, Physics-Driven Learning, Phys. Rev. Appl. **18**, 014040 (2022).

[31] F. Patino-Ramirez, A. Boussard, C. Arson, and A. Dussutour, Substrate composition directs slime molds behaviour, Sci. Rep. **9**, 15444 (2019).

[32] A. Boussard, A. Fessel, C. Oettmeier, L. Briard, H. G. Döbereiner, and A. Dussutour, Adaptive behaviour and learning in slime moulds: The role of oscillations, Philos. Trans. R. Soc. B **376**, 20190757 (2021).

[33] H. Yamada, A. Toth, and T. Nakagaki, Intelligence: Maze-solving by an amoeboid organism, Nature (London) **407**, 470 (2000).

[34] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki, Rules for biologically inspired adaptive network design, Science **327**, 439 (2010).

[35] A. Tero, K. Yumiki, R. Kobayashi, T. Saigusa, and T. Nakagaki, Flow-network adaptation in *Physarum amoebae*, Theory Biosci. **127**, 89 (2008).

[36] K. Alim, N. Andrew, A. Pringle, and M. P. Brenner, Mechanism of signal propagation in *Physarum polycephalum*, Proc. Nat. Acad. Sci. USA **114**, 5136 (2017).

[37] M. Kramar and K. Alim, Encoding memory in tube diameter hierarchy of living flow network, Proc. Natl. Acad. Sci. USA **118**, e2007815118 (2021).

[38] We acknowledge Nachi Stern for suggesting the use of the word *modality*.

[39] R. A. Fisher, Iris. UCI Machine Learning Repository, 1988. DOI: 10.24432/C56C76.

[40] Note that d may not be equal to degree of the node: given two neighboring nodes A and B, it is possible that B is in the d nearest neighbors of A, while A is *not* in the d nearest neighbors of B.

[41] Wine, UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/wine, 1991.

[42] Here L^{-1} is the Moore Penrose pseudoinverse of L. Note that L is not invertible.

[43] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of nNeural System*, Ch. 8 (MIT Press, Cambridge, MA, 2005).

[44] M. Stern, S. Dillavou, M. Z. Misken, D. J. Durian, and A. J. Liu, Physical learning beyond the quasistatic limit, Phys. Rev. Res. **4**, L022037 (2022).

[45] A. Koseska and P. I. Bastiaens, Cell signaling as a cognitive process, EMBO J. **36**, 568 (2017).

[46] C. J. Chan, A. E. Ekpenyong, S. Golfier, W. Li, K. J. Chalut, O. Otto, J. Elgeti, J. Guck, and F. Lautenschläger, Myosin II activity softens cells in suspension, Biophys. J. **108**, 1856 (2015).

[47] A. Laskar, R. K. Manna, O. E. Shklyaev, and A. C. Balazs, Computer modeling reveals modalities to actuate mutable, active matter, Nat. Commun. **13**, 2689 (2022).

[48] M. Stern, C. Arinze, L. Perez, S. E. Palmer, and A. Murugan, Supervised learning through physical changes in a mechanical system, Proc. Natl. Acad. Sci. USA **117**, 14843 (2020).

[49] E. L. Schwartz, *Computational Neuroscience* (MIT Press, Cambridge, MA, 1993).

[50] K. P. Körding and P. König, Supervised and unsupervised learning with two sites of synaptic integration, J. Comput. Neurosci. **11**, 207 (2001).

[51] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[52] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, Mastering the game of go without human knowledge, Nature (London) **550**, 354 (2017).

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, Attention is all you need, in *Adv. Neural Inf. Process. Syst.* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), Vol. 30, (Curran Associates, Inc., New York, 2017).

[54] A. Y. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng, Deep speech: Scaling up end-to-end speech recognition, arXiv:1412.5567.

[55] For simplicity we assume that the chemical has negligible diffusion and the only way it can spread is via the network currents.

[56] The current flows into and out of the network through boundary nodes.

[57] This helps the potentials at the output node to converge nicely at the desired potentials. If this delay is not introduced, the output potential oscillates about the desired potential. Moreover, the delay helps in avoiding the buildup of excess chemical in the network.