APL Machine Learning

RESEARCH ARTICLE | FEBRUARY 27 2024

Training self-learning circuits for power-efficient solutions

Menachem Stern ; Sam Dillavou ; Dinesh Jayaraman ; Douglas J. Durian ; Andrea J. Liu

APL Mach. Learn. 2, 016114 (2024)
https://doi.org/10.1063/5.0181382

Training self-learning circuits for power-efficient solutions

Cite as: APL Mach. Learn. 2, 016114 (2024); doi: 10.1063/5.0181382
Submitted: 16 October 2023 • Accepted: 1 February 2024 •
Published Online: 27 February 2024

Menachem Stern,[1,a)] Sam Dillavou,[1] Dinesh Jayaraman,[2] Douglas J. Durian,[1,3] and Andrea J. Liu[1,3]

AFFILIATIONS

[1] Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA
[2] Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA
[3] Center for Computational Biology, Flatiron Institute, Simons Foundation, New York, New York 10010, USA

[a)] **Author to whom correspondence should be addressed:** nachis@sas.upenn.edu

ABSTRACT

As the size and ubiquity of artificial intelligence and computational machine learning models grow, the energy required to train and use them is rapidly becoming economically and environmentally unsustainable. Recent laboratory prototypes of self-learning electronic circuits, such as "physical learning machines," open the door to analog hardware that directly employs physics to learn desired functions from examples at a low energy cost. In this work, we show that this hardware platform allows for an even further reduction in energy consumption by using good initial conditions and a new learning algorithm. Using analytical calculations, simulations, and experiments, we show that a trade-off emerges when learning dynamics attempt to minimize both the error and the power consumption of the solution—greater power reductions can be achieved at the cost of decreasing solution accuracy. Finally, we demonstrate a practical procedure to weigh the relative importance of error and power minimization, improving the power efficiency given a specific tolerance to error.

I. INTRODUCTION

There has been a meteoric rise in the adoption and usage of artificial intelligence (AI) and machine learning (ML) tools in just the past 15 years,[1,2] accompanied by an equally spectacular rise in the sizes of ML models and the amount of computation required to train and apply them.[3,4] In recent years, the energy required to train state-of-the-art ML models, as well as to use the trained models, has been rising exponentially, doubling every 4–6 months.[5] This energy cost will eventually severely constrain further increases in model complexity and already constitutes significant economic and carbon costs.[6–8]

The field of neuromorphic computing[9–13] strives to recreate the ability to learn in hardware. A major motivation for the development of neuromorphic systems is the possibility of massive energy savings compared to ML implemented on standard computers.[13,14] Many proposals for synthetic "neurons" and "synapses" have been laid out over the past three decades, promising lower power consumption compared to standard computers by 2–5 orders of magnitude.[15–18] While much neuromorphic computing research has been focused on the development of power-efficient hardware, usually for performing inference (applying already-trained ML models), some attention has recently been given to the study of power-efficient learning "algorithms."[19–22] However, most neuromorphic hardware implementations considered thus far specifically attempt to mimic standard ML algorithms, such as backpropagation[23–25] or phenomenological neural synaptic learning processes such as STDP (spike-timing-dependent plasticity).[26–30]

Recently, a new avenue was opened toward realizing power-efficient neuromorphic computing, dubbed *physical learning machines* or *self-learning physical networks*.[31] Rather than mimicking known learning algorithms, such as backpropagation, such systems exploit their inherent physics in order to learn, using *local learning rules* that modify the *learning degrees of freedom* based on locally available information, such that the system globally learns to perform desired tasks. A certain class of local learning rules, known as *contrastive learning*,[32–36] describe how learning degrees of freedom should be modified in order for systems to achieve desired outputs in response to inputs supplied by observed examples of use (i.e., supervised learning).

In order to realize any power gains, such learning rules must be implemented in hardware. *Coupled learning*, a particular contrastive learning rule, has been realized successfully in laboratory hardware for electronic circuits of variable resistors.[37–40] Such systems already consume less power than conventional computers doing inference because they are analog rather than digital.[41] Here, we use analytical theory, computation, and experiments to show that the propensity of electronic circuits to minimize power dissipation enables even greater reductions in power consumption via appropriate initialization and power-efficient learning rules. We specifically demonstrate these results for regression tasks. However, it should be noted that our analysis and results should apply to other physical learning machines in different physical media (e.g., mechanical networks) if they can be developed in the lab,[42] as well as to other types of problems (e.g., classification).

This paper is organized as follows: in Sec. II, we describe the physical learning approach and discuss how the power consumption of the system is modified by learning, in particular, as we change the initial conditions of the learning degrees of freedom. A judicious choice of initial conductances yields learning solutions with low power consumption, while also reducing the energy consumed in training. In Sec. III, we introduce a modification to the local physical learning rule in order to minimize both error and power consumption. We analyze this new local rule theoretically and test it in simulations and lab experiments, concluding that it leads to an error–power trade-off; lower-power solutions may be obtained at the expense of higher errors. The energy required to train the system can be reduced as well. Finally, in Sec. IV, we demonstrate how a power-efficient learning algorithm with dynamical control over the weighting of power and error optimization can lead to an efficient adaptation of low-power solutions beyond simply using good initial conditions and constant weighting.

II. POWER CONSUMPTION IN PHYSICAL LEARNING CIRCUITS

In previous work, we established theoretically and experimentally that self-learning resistor networks can be trained to perform tasks such as allostery, regression, and classification.[37,39,40] Training a deep neural network corresponds to minimizing a learning cost function with respect to learning degrees of freedom (edge weights and biases). The learning landscape, described by the learning cost function as one axis in the high-dimensional space where each of the other axes corresponds to a different learning degree of freedom, remains fixed during the minimization. On the other hand, successful training of physical learning machines corresponds to the simultaneous minimization of *two* cost functions—the learning and physical cost functions—with respect to two different sets of degrees of freedom (DOF), the learning and physical degrees of freedom, respectively. In the case of a self-learning electrical network of variable resistors, the physical cost function is the dissipated power, the physical DOF are the node voltages, and the learning DOF are the conductances.

Notably, the physical cost function, or power, depends implicitly on the learning DOF. As a result, both the learning landscape and the physical landscape *evolve* during training. For example, training gives rise to soft modes in the physical landscape and to stiff modes in the learning landscape, making the system more conductive and lowering its effective response dimension.[43]

The height of a minimum in the physical landscape corresponds to the power required to actuate the desired response (to obtain the desired outputs in response to the given inputs from training data). Due to the coupling between the learning and physical landscapes, it is possible to find and push down the minima in the physical landscape corresponding to the global minima in the learning landscape during training, thus decreasing the amount of power required to perform a given task.

Consider an electrical circuit that minimizes a scalar physical cost function $P(V;\, k)$ (e.g., the dissipated power), depending on a set of physical DOF V (e.g., the node voltages) and a set of learning DOF k (e.g., the edge conductances). When an input signal (e.g., a set of voltages at input nodes) is applied, the system responds by optimizing the physical DOF to minimize P subject to the input constraints, producing a stable free state V^F with an associated free power $P^F(V^F;\, k)$. Training this system for specific output responses using coupled learning[34] involves clamping the targets T by slightly nudging them toward the desired response $V_T^C = V_T^F + \eta(\tilde{V}_T - V_T^F)$, with $\tilde{V}_T$ being the desired response and nudge amplitude $\eta \ll 1$. The physical system then minimizes the physical cost function subject to both the inputs and this clamping, yielding a clamped state V^C with a clamped power $P^C(V^C;\, k)$. The contrast (or contrastive function) is defined as the difference between the physical cost (powers) for the clamped and free states,

$$\mathcal{C} \equiv \eta^{-1}[P^C - P^F], \tag{1}$$

which is intrinsically non-negative. Minima with vanishing contrast are also minima of the error (loss) function $\mathcal{L}$ that is typically used to measure the quality of a learning solution, e.g., the mean squared difference between the desired and obtained behavior, $\mathcal{L} \equiv \frac{1}{2}(\tilde{V}_T - V_T^F)^2$.[34]

Physical learning is achieved by a learning rule that corresponds to modifying the learning degrees of freedom according to the partial derivative of the contrast. This learning rule is local,

$$\dot{k} = -\alpha \frac{\partial \mathcal{C}}{\partial k_i} = -\alpha \eta^{-1} \frac{\partial}{\partial k_i}[P^C - P^F], \tag{2}$$

with α being a scalar *learning rate*, setting the time scale for the learning dynamics. A system following these dynamics with a sufficiently low learning rate tends to minimize the learning cost function $\mathcal{L}$ [as shown in Fig. 1(b)]. See Appendix A for more details on the learning dynamics close to a solution for the learning degrees of freedom k^*.

A. Power consumption in learned solutions

We now turn our attention to the scalar physical cost (i.e., power consumption) of the free state P^F. This *free power* is the relevant measure for the power used by the system to perform inference. As noted, this is the power associated with the application of the input voltages, allowing the output readouts. Our work is primarily concerned with minimizing this free power P^F, while also achieving good learning solutions with low error. We will show how the free power can be lowered by non-hardware means, such as choosing better initialization for the learning DOF and using learning rules that minimize the free power at the same time as the error.

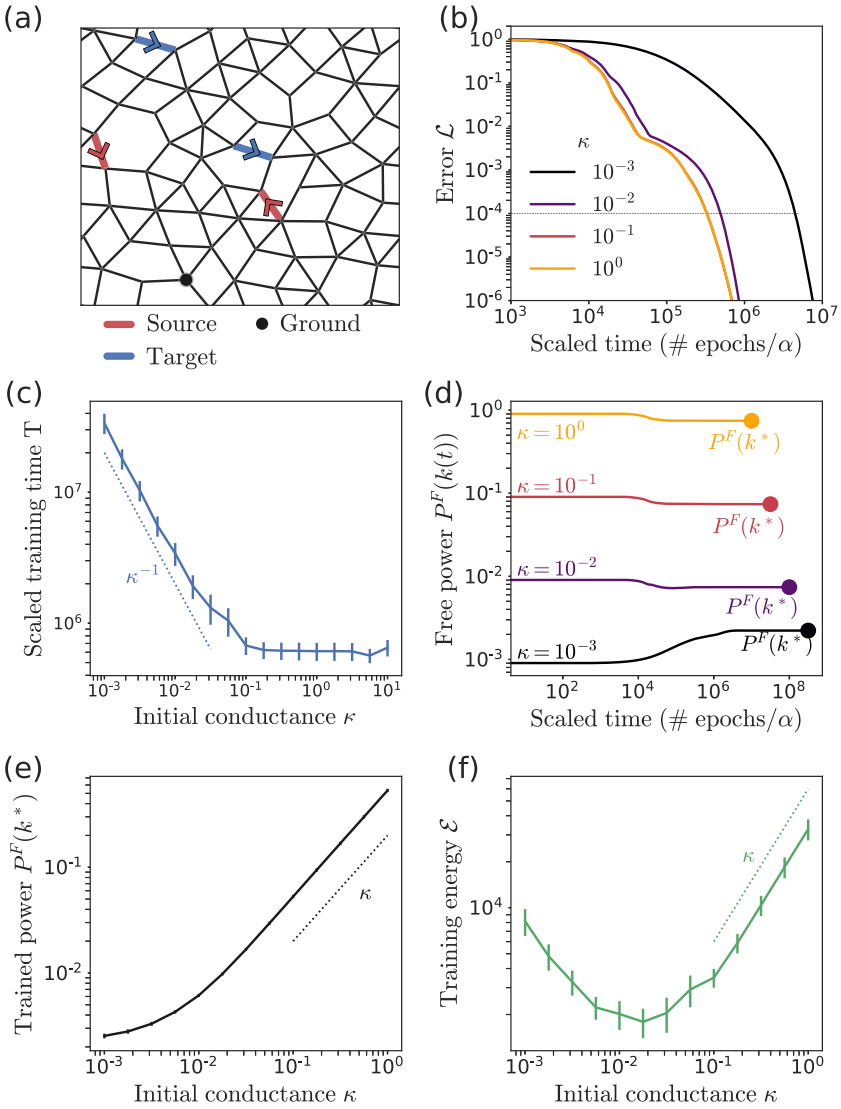

FIG. 1. Effects of varying the conductance initialization scale. (a) Simulated resistor networks with edges corresponding to variable resistors. We train networks with $N = 64$ nodes to perform linear regression, i.e., to simulate desired linear equations with two variables (red source edges) and two results (blue target edges), see Appendix B for details. (b) The error $\mathcal{L}$ as a function of training time for several conductance initialization values κ. The error is successfully reduced by the coupled learning rule by multiple orders of magnitude, regardless of the choice of the initialization scale κ. (c) Training time T (epochs taken for the error to drop to error level $\tilde{\mathcal{L}} = 10^{-4}$) as a function of initialization κ. The training time remains constant when initialization is far from the bounds but grows linearly for low initialization close to $k_{\min}$. (d) Free power P^F during training for different initialization κ. At the end of the training, the system finds a solution with trained power marked by the colored dots. (e) The trained power as a function of initialization κ. Decreasing the conductance initialization scale has a strong effect, reducing the trained power needed to actuate the learned solution. (f) Training energy $\mathcal{E}$ as a function of initialization κ. Choosing a proper optimal initialization (here, $\kappa \approx 2 \times 10^{-2}$) can optimize the training energy. The results averaged over 50 realizations of networks and regression tasks.

The free power of a system trained for a specific task will, henceforth be referred to as the *trained free power*. We will also look at the total energy required to train the system $\mathcal{E}$, henceforth termed as the *training energy*, which can be estimated by integrating the free power over the training time. We show how this training energy can also be optimized by these initialization schemes and learning rules.

We first study how the free power P^F is affected by the basic coupled learning rule of Eq. (2), and later see how it can be substantially reduced by modifying this rule. Using the chain rule in Eq. (2), we can derive an ODE for the free power during training,

$$\dot{P}^F(k) = \dot{k}^T \nabla_k P^F(k) = -\alpha \partial_k \mathcal{C}^T \cdot \partial_k P^F. \tag{3}$$

Note that the free state is a minimum of the power subject to the inputs so that the derivative $\partial_V P^F$ vanishes exactly, and hence, $\nabla_k P^F = \partial_k P^F + \frac{dv}{dk}\partial_V P^F = \partial_k P^F$. We see that the free power tends to decrease if the gradients of the free power and the contrast with respect to k align and increase otherwise. Assuming the free power changes slowly with k, or that the learning DOF k are close to the learning solution k^*, we can approximate the free power using the following Taylor expansion:

$$P^F(k) \approx P^F(k^*) + (k-k^*)^T \partial_k P^F(k^*). \tag{4}$$

The free power changes due to the learning dynamics, starting at the initial condition $P^F(k^0)$ and ending after training with $P^F(k^*)$. Next, we discuss the sign of this shift, determined by the alignment between the gradients of the contrast and free power.

Here, we specialize to the case of learning electrical circuits, e.g., adaptive resistor networks, where the physical DOF are the voltages at nodes V_a, while the learning DOF are conductances k_i of edges i connecting pairs of nodes. An adjacency matrix Δ_{ia} is defined such that each row of the matrix corresponds to an edge, having a value of +1 at the index of the incoming node of that edge, −1 at the index of the outgoing node, and 0 elsewhere. The choice of which node is incoming or outgoing is a matter of convention and sets the direction of currents but has no physical consequence. The vector of voltage drops on the edges is given by $\Delta V_i = \sum_a \Delta_{ia} V_a$. Resistor networks minimize the total power dissipation,

$$P = \frac{1}{2}\sum_i k_i \Delta V_i^2. \tag{5}$$

In such networks, where one of the nodes is grounded at $V_G = 0$, the native state of the network (in the absence of any inputs) is where all voltage values are zero, all voltage drops are zero, and the total power dissipation is $P = 0$. When the free/clamped boundary conditions are applied, for example, by introducing currents in certain input and output edges, the free and clamped power are

$$P^{F,C} = \frac{1}{2}\sum_i k_i (\Delta V_i^{F,C})^2.$$

Given weak clamping $(V^C - V^F \sim \eta \ll 1)$, we write the contrast function $\mathcal{C}$, neglecting the terms of order η^2,

$$\begin{aligned}\mathcal{C} &= \eta^{-1}[P^C - P^F] \\ &\approx \eta^{-1}\sum_i k_i [\Delta(V^C - V^F)]_i [\Delta V^F]_i.\end{aligned} \tag{6}$$

We take the partial derivative of the contrast with respect to k (as done in Ref. 35),

$$\frac{\partial \mathcal{C}}{\partial k_i} \approx \eta^{-1}[\Delta(V^C - V^F)]_i [\Delta V^F]_i. \tag{7}$$

In this simple case, the learning modification is determined by the alignment of each component of the free state response ΔV_i^F with its nudge in the clamped state $(\Delta V^C - \Delta V^F)_i$. In these particular models, we also know that the free power gradient is positive $\frac{\partial P^F}{\partial k_i} = (\Delta V^F)_i^2 \geq 0$. We conclude that if the clamped state nudge aligns with the free state response, the free power tends to decrease. This is sensible as the system has to decrease its conductances to achieve a stronger response required by the clamping. The opposite effect occurs when the nudged response is misaligned with the free state, resulting in increased conductances.

B. Power dependence on initial conditions

In Sec. II A, we established how physical learning affects the system's free power P^F, i.e., its power consumption in the free state. In the following, we consider how the initial conditions of the learning degrees of freedom determine the trained free power, i.e., free power of the learned solutions. We will show that judicious initialization leads to considerable savings in power consumption.

It is well recognized in the ML literature that the dynamics and obtained solutions of learning algorithms strongly depend on initialization, i.e., the initial values of the learning DOF.[44–47] In the context of physical learning, the choice of initialization may not only affect the training time and accuracy of a solution but may also have important effects on the trained free power. Suppose a set of voltage drops is applied over some input edges of a resistor network, and we read out the resulting voltage drops over some other output edges. In addition, suppose that the conductance values of the network have a certain scale κ. It is known that the output voltage drops do not depend on the scale κ but only on the relative ratios of the conductance of different edges. However, reducing the conductance scale does, in fact, linearly decrease the free power [Eq. (5)]. Thus, we can, in principle, improve the trained free power indefinitely by reducing the conductance scale. Realistically, we are bound by experimental considerations: variable conductive elements have minimal conductance values (corresponding to maximal resistance). Furthermore, low conductance necessitates more precise hardware implementations as the network response becomes highly sensitive to small variations in the conductance.

The above-mentioned considerations suggest that initializing the conductance values k (learning DOF) at lower values may yield solutions with lower trained free power. To verify these ideas, we trained $N = 64$ node networks [Fig. 1(a)] for multiple regression tasks with two inputs and two outputs. The *error* for these regression tasks, noted as $\mathcal{L}$, is given by mean squared differences between the desired and the obtained target voltage drops in the training set (Appendix B presents details on the simulated resistor networks and regression tasks). We initialized the conductance values uniformly with different conductance scales in the range $10^{-3} \leq \kappa \leq 10^{1}$. We note that in these simulations, the minimum conductance for any given edge is $k_{\min} = 10^{-3}$, and the maximum conductance is $k_{\max} = 10^{1}$. Learning modifications that attempt to push the conductances out of this range are not performed. The learning rate α has been chosen such that $\kappa = 1$ and $\alpha = 0.1$, a value which typically results in a relatively quick and stable learning performance for these networks and tasks. We find that as long as the learning rate is chosen to be slow enough, the learning dynamics are well behaved and our results scale as expected with the learning rate.

We set the units for these simulations such that the conductance scale $[k]$ is defined as $k = 1$ inside the allowed range of our conductors, and the voltage scale $[V]$ corresponds to the typical highest input values chosen for our regression tasks $V = 1$ (shown in Appendix B). The networks are trained for 10^6 learning iterations of Eq. (2). Each such iteration encompasses an epoch, i.e., the

time taken for the network to observe and respond to all training examples, similar to full-batch gradient descent.[48] Our units of time are scaled by the learning rate $[\tau_{epoch}] = \alpha^{-1}$. With these definitions, the units for the free power are given by $[P^F] = [k][V]^2$. The training energy is given by the free power, integrated over training time, which has units of $[\mathcal{E}] = [\tau_{epoch}][k][V]^2$.

As expected, we find that coupled learning reduces the error by many orders of magnitude [Fig. 1(b)]. We also find that when the learning rate is scaled appropriately $\alpha \propto \kappa^{0.5}$, the scaled training time T (number of epochs taken for the system to reach a certain error threshold $\tilde{\mathcal{L}} = 10^{-4}$, scaled by the learning rate) does not change much for relatively high initialization κ [Fig. 1(c)]. However, initialization close to the lower boundary $k_{\min}$ induces a linear increase in the scaled training time, scaling as κ^{-1}. This increase in the training time is reasonable as a large part of the training modification Δk_i is not performed because it would require the conductances to go below the minimum.

More importantly, at lower initialization scales κ, physical learning finds lower trained free power solutions $P^F(k^*)$ [in Figs. 1(d) and 1(e), note that the colored dots in panel d mark the trained free power]. These results clearly show the benefit of initializing the conductances of edges close to their minimal values in terms of learning power efficient solutions. While so far, we referred to the power necessary to actuate the solution (i.e., the free power P^F), one often needs to consider the total energy required to train the network to adopt this solution, the training energy $\mathcal{E}$. In some applications, this training energy is small compared to the total energy spent on using the system throughout its life cycle. However, when this is not the case, one should consider learning algorithms that reduce the required training energy and the free power. In our simulations, the training energy $\mathcal{E}$ can be measured as the integral over time of the free power during training, until the error reaches a certain tolerable level (e.g., $\tilde{\mathcal{L}} = 10^{-4}$) at time T,

$$\mathcal{E} = \int_0^T P^F(k(t))dt.$$

We find that the training energy $\mathcal{E}$ scales linearly with the initialization at high κ [Fig. 1(f)], similar to the trained free power. However, lowering κ close to $k_{\min}$ actually *increases* the training energy. This is because we can no longer realize gains in the free power [the saturating region shown in Fig. 1(e)], while the training time increases linearly with decreasing κ [Fig. 1(c)]. As a result, the training energy in this regime increases with decreasing κ [Fig. 1(f)]. Thus, there is an optimal value for the initialization κ corresponding to the minimum training energy. We note that in this regime, as the training energy is proportional to the training time, it is inversely proportional to the learning rate so that increasing the learning rate can reduce the energy $\mathcal{E}$. However, this improvement only persists for low enough learning rates, when the learning process is consistent and well behaved. We leave the study of power efficiency of fast learning system for future study.

In machine learning, however, the greatest energy cost is incurred during inference. In our case, this cost is quantified by the trained free power $P^F(k^*)$. We note that training reduces the free power for high κ but increases it for low κ next to the lower conductance limit [Fig. 1(d)]. This is sensible because for low initial conductances at or near the minimum, the network must increase some edge conductances in order to decrease its error. That being said, we conclude that initializing the network with proper low conductance values can save a significant energy during learning and when using the trained network.

III. EXPLICIT POWER MINIMIZATION

We have seen that the learning rule in Eq. (2) modifies the free power during learning. Our next step is to find a way to explicitly control the learning process to produce solutions with lower trained free power. This is possible because the learning rule in Eq. (2) is already written in terms of the free power. It is natural to modify this learning rule to locally minimize the free power and the error. Consider the addition of an explicit free power minimization term to the contrast,

$$\mathcal{C}_\lambda = \eta^{-1}[P^C - P^F] + \eta^{-1}\lambda P^F, \tag{8}$$

where λ is a tunable parameter, termed as the *power minimization amplitude*, which dictates the importance of free power minimization. The learning rule, the partial derivative of the contrast, then becomes

$$\dot{k} = -\alpha\partial_k \mathcal{C}_\lambda = -\alpha\eta^{-1}\partial_k[P^C - (1-\lambda)P^F]. \tag{9}$$

Note that as the free power can be partitioned as a sum over the network edges, the power minimizing rule is still local and physically realizable. This modified learning rule tends to decrease the free power P^F as the modified learning dynamics lower the free power *and* the contrast in Eq. (1). If we set $\lambda = 1$, the free power cancels out and we recover the directed aging learning rule[49,50] that solely tends to reduce the clamped power.

Using the modified learning rule [Eq. (9)], one can derive ODEs for the contrast and free power, similar to Eq. (3),

$$\begin{aligned}\dot{\mathcal{C}}(k) &= -|\partial_k \mathcal{C}|^2 - \lambda\partial_k \mathcal{C}^T\partial_k P^F,\\ \dot{P}^F(k) &= -\partial_k \mathcal{C}^T\partial_k P^F - \lambda|\partial_k P^F|^2.\end{aligned} \tag{10}$$

These dynamics tend to reduce the value of the contrast $\mathcal{C}$ over time, up to interference from a term that encodes the alignment between the gradient of the contrast and the free power. Moreover, we find that the free power P^F tends to be reduced by these dynamics, again up to an effect determined by the alignment. We now discuss the dynamics of the contrast and free power in a simplified linear setting. First, note that in the limit $\lambda \to \infty$, the learning rule solely minimizes the free power. We denote this free power minimum as k^*_∞. Around this local minimum, the free power can be expanded to quadratic order,

$$P^F(k) \approx P^F(k^*_\infty) + \frac{1}{2}(k - k^*_\infty)^T H(k - k^*_\infty),$$

where $H \equiv \partial_k^2 P^F(k^*_\infty)$ is the free power Hessian with respect to learning degrees of freedom. We can similarly expand the contrast in series around the $\lambda = 0$ learning solution (the unmodified solution discussed earlier),

$$\mathcal{C}(k) \approx \frac{1}{2}(k - k^*_0)^T \mathcal{H}(k - k^*_0),$$

where $\mathcal{H} \equiv \partial_k^2 \mathcal{C}(k_0^*)$ is the contrast Hessian with respect to learning degrees of freedom at $\lambda = 0$ [in over-parameterized networks, the constant term $\mathcal{C}(k_0^*)$ vanishes, see Appendix A]. If the learning solution at finite λ, k_λ^* is close to the limiting solutions k_∞^* and k_0^*, we can express the new contrast approximately as

$$\mathcal{C}_\lambda(k) \approx \frac{1}{2}(k-k_0^*)^T \mathcal{H}(k-k_0^*) + \lambda\left[P^F(k_\infty^*) + \frac{1}{2}(k-k_\infty^*)^T H(k-k_\infty^*)\right]. \tag{11}$$

We can now discuss the dynamics of the learning degrees of freedom $\dot{k} = -\partial_k \mathcal{C}_\lambda$. Taking the partial derivative of Eq. (11), we find a first order ODE for k, whose solution is exponential,

$$k(t) = k_\lambda^* + e^{-(\mathcal{H}+\lambda H)t}[k(t=0) - k_\lambda^*], \quad k_\lambda^* = (\mathcal{H}+\lambda H)^{-1}[\mathcal{H}k_0^* + \lambda H k_\infty^*]. \tag{12}$$

Starting from an initial condition $k(t=0) \equiv k^0$, the learning DOF exponentially decay to k_λ^*. Let us discuss the learning DOF solution k_λ^*. It is clear that when no power minimization is applied, $k_\lambda^* = k_0^*$. If both Hessians $\mathcal{H}, H$ are full rank (and invertible), the λ parameter would smoothly interpolate between k_0^* and k_∞^*. However, we know that the Hessian of the contrast in over-parameterized learning machines is low-rank (with the number of non-zero eigenvalues equal to the number of training tasks, see Appendix A for details).[43] This means that the contrast Hessian $\mathcal{H}$ is not invertible and has vanishing eigenvalues. In the eigen-directions of these vanishing eigenvalues, the power minimization is dominant for any finite value of λ. Thus, the power minimization term introduces a singular perturbation so that for infinitesimal power minimization amplitude $\lambda = 0^+$, the learning solution approaches $k_{0^+}^* = \lim_{\lambda\to 0} k_\lambda^* \neq k_0^*$. The solution $k_{0^+}^*$ tends to minimize the free power, while keeping the contrast low. For over-parameterized learning in the $\lambda \to 0$ limit,

$$k_{0^+}^* = \arg\min_k \; P^F(k) \quad \text{s.t.} \quad \mathcal{C}(k) = 0. \tag{13}$$

The solution k_λ^* is then a weighted average of the limiting solutions $k_{0^+}^*, k_\infty^*$, weighted by the Hessian matrices $\mathcal{H}, \lambda H$. For weak power optimization ($\lambda \ll 1$),

$$k_\lambda^* \approx k_{0^+}^* + \lambda s, \quad s \equiv (\mathcal{H}+\lambda H)^{-1} H(k_\infty^* - k_{0^+}^*). \tag{14}$$

For $\lambda \ll 1$, note that the vector s is nearly constant as the inverse matrix is dominated by $\mathcal{H}$. This means the solutions shift λs is approximately linear in the optimization parameter λ (see Appendix A). Let us further denote $\Delta k^0 = k^0 - k_\lambda^*$ and introduce a time propagator $U_\lambda(t) \equiv e^{-(\mathcal{H}+\lambda H)t}$. The solution for k can be plugged in the equations above to express the dynamics of the contrast and free power,

$$\mathcal{C}(t) \approx \frac{1}{2}\Delta k^{0T} U_\lambda \mathcal{H}[U_\lambda \Delta k^0 + 2\lambda s] + \frac{1}{2}\lambda^2 s^T \mathcal{H} s,$$
$$P^F(t) \approx P_{0^+}^F + \frac{1}{2}\Delta k^{0T} U_\lambda [H U_\lambda \Delta k^0 + 2\partial_k P_{0^+}^F] - \lambda(\partial_k P_{0^+}^F)^T(\mathcal{H}+\lambda H)^{-1}[H U_\lambda \Delta k^0 + \partial_k P_{0^+}^F]. \tag{15}$$

For both the contrast and free power, we keep the largest non-vanishing contribution at long times due to the modified learning dynamics,

$$\mathcal{C}(t\to\infty) \approx \frac{1}{2}\lambda^2 s^T \mathcal{H} s, \quad P^F(t\to\infty) - P_{0^+}^F \approx \lambda(\partial_k P_{0^+}^F)^T(\mathcal{H}+\lambda H)^{-1}\partial_k P_{0^+}^F. \tag{16}$$

These are our key results: (1) the error induced by power minimization scales with λ^2, while (2) the free power reduction compared to $P^F(k_{0^+}^*)$ scales linearly with λ. In other words, the *free power difference*, i.e., the difference between the free power with and without λ-minimization, $\Delta P_\lambda^F \equiv P_\lambda^F(k_\lambda^*) - P_{0^+}^F(k_{0^+}^*) \propto \lambda$, scales linearly with this minimization amplitude.

Our argument considers the error and trained free power of solutions at infinite training time but a practical learning scenario ends after some finite training time $t = \tau$. This training time must be large compared to the natural scale of the contrast Hessian to allow learning to occur. However, in the weak power minimization limit, this time can be much smaller than the power minimization timescale $\mathcal{H} \gg \tau^{-1} \gg \lambda H$. In this case, the dynamics can be approximated by a fast decay toward the unmodified solution k_0^*, followed by a slow decay from k_0^* to the power minimizing solution k_λ^*. For small λ, the learned solution at a finite time τ is

$$k(\tau) - k_0^* \approx \lambda\tau \times H(k_\lambda^* - k_0^*). \tag{17}$$

The learned solution moves away from k_0^* at a rate linearly proportional to λ. This solution can be used to estimate both the contrast

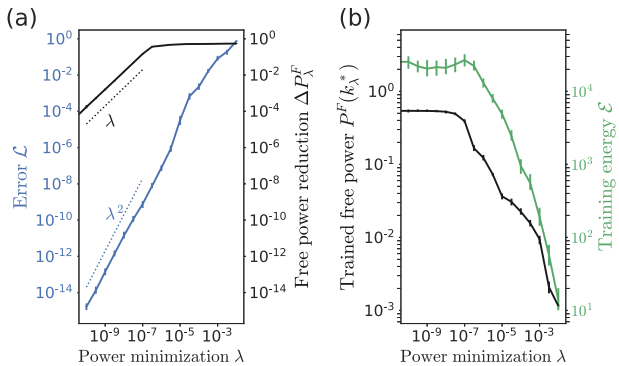

FIG. 2. Physical learning with power minimization. (a) Error $\mathcal{L}$ (blue) and trained free power difference between learning with and without power minimization, ΔP_λ^F (black), for varying values of the power minimization amplitude λ. As λ is increased, the error of the learned solution increases quadratically but the trained free power of these solutions is linearly decreased. (b) The trained free power (black) as well as the training energy $\mathcal{E}$ (green) decreases with λ, underscoring a trade-off between power-efficiency and error. The results are averaged over 50 realizations of networks and training tasks.

and the free power at time τ. As seen before, we find that the contrast scales as λ^2, while the free power is reduced proportional to λ and the elapsed time $\Delta P_\lambda^F \sim \lambda\tau$.

Overall, these considerations suggest that under λ-modified dynamics, a trade-off emerges between the error and trained free power. This intuition is verified in the numerical simulations shown in Fig. 2. We train a 64 node resistor network, initialized with intermediate conductance values $k_i^0 = 1$, for a regression task as before. Here, the training proceeds with the modified power minimization learning rule [Eq. (9)], varying λ in the range $10^{-10} \leq \lambda \leq 10^{-2}$. We train these networks for $\tau = 10^5$ steps and then measure the trained error and free power, averaging the results over 50 realizations of the network and regression tasks. We find that for small λ, the error $\mathcal{L}$ and free power reduction ΔP_λ^F scale as predicted by Eq. (16) [Fig. 2(a)]. As before, we also compute the training energy $\mathcal{E}$. We plot the trained free power $P^F(k_\lambda^*)$ and the training energy as a function of the minimization amplitude λ in Fig. 2(b). Both of these are markedly decreased when λ is increased, showing the predicted trade-off between power efficiency and error. In these settings, choosing the optimization parameter $\lambda = 10^{-5}$ allows us to maintain a reduction of five orders of magnitude in error, while reducing the free state power and the total energy required for training by a factor ~10 compared to standard learning ($\lambda = 0$).

A. Experimental results

So far, we argued on theoretical grounds that error can be traded-off for power efficiency by employing the learning rule in Eq. (9) and verified these arguments in simulations. Here, we verify the existence of the trade-off in laboratory experiments. We use an experimental network of variable resistors implementing coupled learning, similar to the realizations in previous studies.[37–39] However, in this new implementation of the experiment, transistors replace the digital potentiometers in the role of variable resistors.[41] Unlike in the previous work,[37] this system is also able to learn according to the continuous coupled learning rule [Eq. (2)] as each resistance element is set by a charged capacitor on the gate of the transistor instead of by a discrete counter. Modifications to the

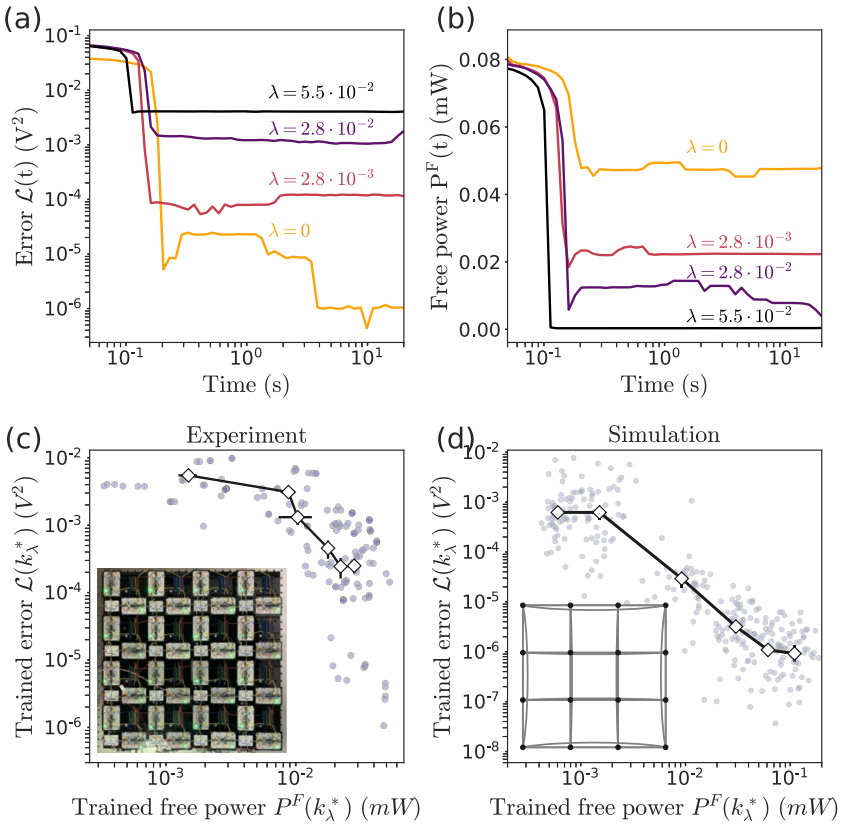

FIG. 3. Experimental results for power optimization show a trade-off between error and power. (a) Error $\mathcal{L}$ as a function of time in laboratory experiments with different optimization amplitude values λ. An adaptive nonlinear resistors network can physically learn to adopt the desired function. This network learns to perform node allostery tasks, gradually minimizing the error down to a finite error floor. (b) Free power in physical learning experiments for different values of λ. As experiments are run with increasing λ, the learning process finds solutions with an increasing error but with improved trained free power. (c) Error vs the trained free power of experimentally learned solutions. Overall, we observe an error–power trade-off in this experimental learning machine. The inset shows a photograph of the experiment. (d) Error vs the trained free power of learned solutions in numerical simulations on the same network geometry and type of tasks. The trade-off between power efficiency and error is recapitulated in simulated learning resistor networks, where the units of time conductance and voltage are matched with the experiments.

learning rule of the form in Eq. (9) are achieved by varying the measurement amplification from the free and clamped networks. In addition, unlike previous implementations, this new network operates continuously in time, with the clamped state value updated automatically via an electronic feedback loop, and so, training duration is measured in real time rather than training steps. Because of unavoidable noise in the experiment, $\eta \to 0$ is unobtainable. As the clamped state approaches the free state, their difference becomes more and more difficult to measure. Therefore, we use a finite value $\eta = 0.22$ for these experiments with an effective learning rate of $\alpha = \frac{1}{24\ \text{ms}}$. The experiments lasted 20 seconds each, and the network's resistances had completely settled at the end of each run. The network is a 4×4 square lattice of edges [inset in Fig. 3(c)] with periodic boundary conditions; the edges are initialized with uniform conductance in the approximate middle of their range at the start of each experiment.

The network was trained for 150 two-source, two-target node allostery tasks, wherein the sources were held at the low and high ends of the allowable range (0 and ~0.45 V, respectively), with the two desired target outputs at either 20% and 80% or at 10% and 90% of this range, respectively. Across these experiments, λ was varied to seven values ranging from 0 to 0.055. In all cases, the network was able to lower the error, as shown for typical error vs training time curves in Fig. 3(a). For these tasks, the network also consistently lowered the free power, as shown for the complementary power curves over training time in Fig. 3(b). Consistent with theoretical predictions, error and trained power increased and decreased, respectively, with increasing λ, with their trade-off shown in Fig. 3(c). White diamonds correspond to the mean error and the trained free power of all the experiments performed with the same value of λ.

To study this trade-off seen in the experiment, we simulated $N = 16$ node resistor networks constructed similarly to the experimental network [the inset in Fig. 3(d)]. These networks are simulated for similar two-source, two-target node allostery tasks (see Appendix B). We added a Gaussian white noise term to Eq. (9) with scale $\delta = 10^{-3}V^2$ to approximate the noisy conditions of experimental learning. The white noise term leads to an error floor $\mathcal{L} \sim 10^{-6}V^2$. A time step in our simulations is equivalent to one experimental learning step (~0.1 ms), while we can set the simulated conductance and voltage scales to match the experiment as well (conductance scale $[k] = 10^{-3}\ \Omega^{-1}$ and voltage scale $[V] = \text{Volt}$). The results for error and free power with λ in the range 10^{-6}–10^{-3}, averaged over 50 realizations of the network and tasks, are shown in Fig. 3(d) and qualitatively show the same error–free power trade-off. However, we note that these simulations are not intended as faithful realistic representations of the experimental learning machine as we simulate a linear flow network and are not attempting to model the specific details regarding the noise and bias profiles of the experiment. The comparison here is only intended to show that realized experimental learning machines display a qualitatively similar performance to power efficiency trade-off as predicted by our theory.

IV. DYNAMICAL CONTROL FOR GREATER POWER MINIMIZATION

In Sec. III, we showed how adding an explicit power minimization term in the contrast function leads to a new local learning rule that attempts to minimize both the error and the free power at the same time, leading to a trade-off between them controlled by the power minimization amplitude λ. We note that noisy inputs make it impossible to reach zero training error, and in any case, there is experimental noise in the self-learning circuits so there is, in practice, a non-zero error floor. Here, we use this insight to design a practical control scheme to dynamically modify λ during learning in order to attain a tolerable error with more power-efficient solutions. We will show how such a control scheme can yield even more power-efficient solutions compared to using a smart initialization (as in Sec. II) and constant λ (as in Sec. III).

Assume that we initialize the conductances of a resistor network at their minimal value (maximum resistance). This initialization leads to a free state $V^F(k_{\min})$ with the lowest possible free power $P^F_{\min}$. This state corresponds to the minimum power found by the power minimization dynamics with $\lambda \gg 1$, which selects the learning degrees of freedom resulting in the lowest free power $P^F_{\min}$. As seen in Fig. 2, reducing the amplitude λ from infinity toward zero monotonically decreases the error, while increasing the trained free power $P^F(k^*_\lambda)$.

Here, we consider a simple dynamical control scheme. Briefly, we set a specific error tolerance as a target, $\tilde{\mathcal{L}}$. We measure the instantaneous error $\mathcal{L}$ while learning using the local rule in Eq. (9). If the instantaneous error is larger than the desired tolerance, we decrease λ to promote error minimization, while if the error is smaller than the tolerance, we increase λ to emphasize power minimization. In other words,

$$\dot{\lambda} = \rho^{-1}\left[\left(\frac{\tilde{\mathcal{L}}}{\mathcal{L}}\right)^p - 1\right]\lambda, \tag{18}$$

with ρ setting the update timescale of λ and the parameter p controlling the rate of the control scheme (a low p value sets the first term in the parentheses close to 1 so that λ dynamics are slow).

To test this dynamical control scheme for learning with power minimization, we simulate the training of $N = 64$ nodes for regression tasks as before. We initialize the conductance values at their minimum $k_{\min} = 10^{-3}$ and set $\alpha = 0.03, \rho = 1, p = 0.02$. We find that the network trained with the λ dynamical control scheme quickly converges on the desired error tolerance [Fig. 4(a), full line and closed circle. We compare these results with an "early stopping algorithm," defined as follows: In this algorithm, we consider a learning network without power minimization ($\lambda = 0$) [the dashed line shown in Fig. 4(a)]. The network reaches the desired error tolerance $\tilde{\mathcal{L}} = 10^{-3}$ after some time [marked by the open circle on the dashed line in Fig. 4(a)], which we call the "early stopping time." Note that our dynamical control scheme in Eq. (18) reaches the same error at a time given by the solid circle on the solid line. Evidently, the dynamical control scheme achieves a lower trained free power compared with early stopping [Fig. 4(b)]. Once the dynamical control scheme reaches the time indicated by the solid circle in [Fig. 4(b)], λ stays constant but the system now keeps training at this value of λ, finally reaching a steady state at long times. As a result, the power advantage of this scheme [gray arrow in Fig. 4(b)] improves over training time until it converges at some free power value.

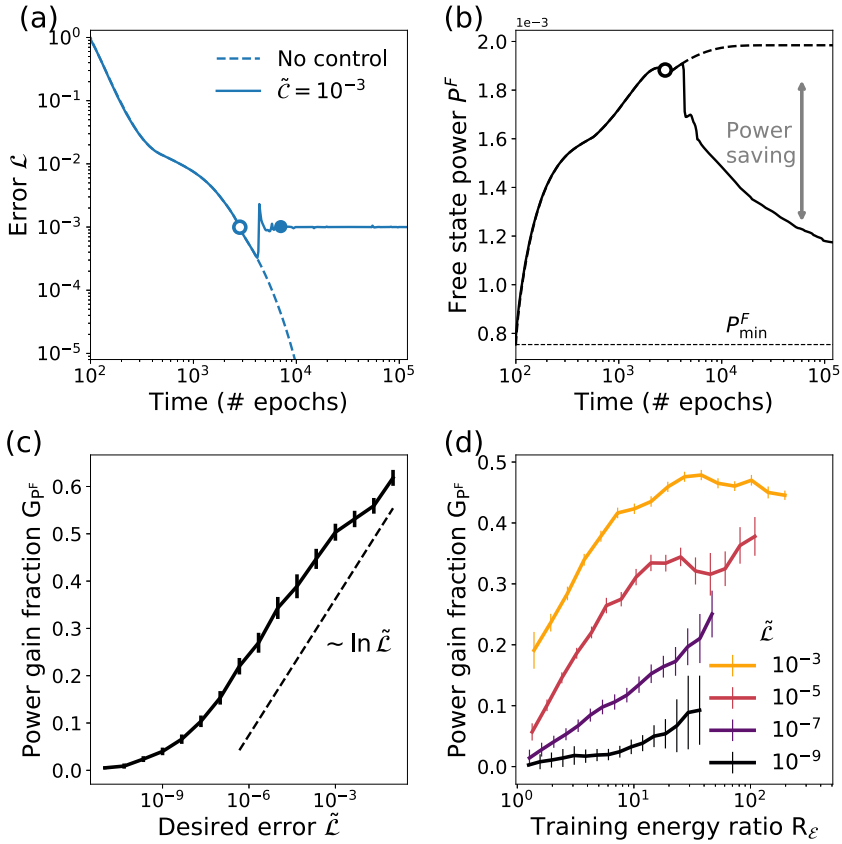

FIG. 4. Power-efficient solutions with dynamical control. (a) Error trajectories with our λ dynamical control scheme (full line) compared to simple learning without power minimization (broken line). We see that the controlled learning rapidly converges to a desired tolerance error level of $\mathcal{L} = 10^{-3}$. (b) Free power under dynamical control of λ vs free power without power minimization. The dynamically controlled system finds solutions that lower the free power compared to an early stopped training of the uncontrolled system at $\mathcal{L} = 10^{-3}$ (open dot). The gray arrow signifies the saved power. (c) The power gain given our control scheme G_{P^F} compared to early stopping for different levels of tolerable error $\tilde{\mathcal{L}}$. We find that dynamical control can generate significantly more power-efficient solutions. (d) Power gain G_{P^F} compared to the ratio of training energies $\mathcal{R}_\mathcal{E}$ between the dynamically controlled learning and early stopping algorithm. We find that to utilize the full benefit of low free power solutions, one needs to train the system for longer times, increasing the network training energy. All results are averaged over 50 realizations of networks and tasks.

We measure the power gain fraction

$$G_{P^F} \equiv \frac{P^F_{\tilde{\mathcal{L}}} - P^F_{\min}}{P^F_{\text{EarlyStop}} - P^F_{\min}}$$

at long training times and compare to the trained free power for the early stopping algorithm for different error tolerances [Fig. 4(c)]. The power saving fraction is measured at $\tau = 10^5$ in relation to the minimal free power produced by the network given for the lowest possible conductance values $k_{\min}$. As higher error $\tilde{\mathcal{L}}$ is tolerated, the dynamical control scheme improves in comparison to the simple early stopping algorithm, saving an additional fraction of power that scales as $\ln \tilde{\mathcal{L}}$. In this case, tolerating an error level of $\tilde{\mathcal{L}} = 10^{-3}$ allows us to save ~50% of the trained free power P^F required for inference. We emphasize that this improvement in power is on top of using the best conductance initialization.

However, we note that gaining the full benefit of this power reduction requires long training, possibly much longer than the early stopping time, meaning that the training energy $\mathcal{E}$ is higher compared to the early stopping algorithm. This consideration means that in our dynamical control scheme, there is a trade-off between the training energy and the trained free power of the solution. This is verified in Fig. 4(d), where we measure the power gain G_{P^F} in terms of the total training energy ratio between the dynamical control scheme and early stopping algorithm, $R_\mathcal{E} = \mathcal{E}/\mathcal{E}_{\text{EarlyStop}}$. Such trade-off also depends on the error tolerance, but we find that if one is willing to spend ~5 – 10 times the training energy compared to the early stopping algorithm, the network achieves most of the benefit of power reduction due to the dynamical control scheme. If the training energy is a major concern and constitutes a significant fraction of the energy expended by the network during its life, one should consider this trade-off for overall lowest power solutions. Finally, we note that our dynamical control scheme is not optimized. Choosing

different parameters or another dynamical control scheme altogether may produce superior power saving at a possibly lower training energy.

V. DISCUSSION

In this work, we studied how electrical circuits can physically learn to adopt desired functions in power-efficient ways. We established that physical learning affects the free power required to actuate the circuit given input signals. This free power can be lowered by choosing better initialization schemes for the learning degrees of freedom, e.g., initializing low conductances in electronic resistor networks.

We have also introduced a modified local learning rule that attempts to minimize both the error and the free power. We showed that this learning rule indeed lowers the trained free power of the obtained learning solutions in both simulations and experiments. This learning rule weights the importance of minimizing error vs power, giving rise to a trade-off between the two. While improving power efficiency at the expense of error (performance) may seem undesirable, a very low error is typically not required and can even be infeasible in real learning situations. Therefore, one can often train learning networks for lower power solutions without much of an adverse effect (Appendix C). In our experiments, there is a natural noise floor and there is no point in striving for a lower error than the floor. For these systems, power-efficient learning rules can improve the solution power with little to no penalty in error.

Finally, we have introduced a dynamical scheme for controlling the relative importance of error and power minimization to rapidly converge on power-efficient solutions with desired error tolerance. We find that such dynamical control can lead to lower power solutions. It is likely that an optimized version of such a dynamical control scheme could further reduce both the solution power and the overall training energy. This is a subject of future study.

While we presented details of the analytical approach for the case of resistor networks, our theoretical arguments apply to other physical systems trained using coupled learning, such as mechanical spring networks (Appendix D). Neuromorphic computing often promises to improve power efficiency by embedding learning algorithms in hardware, solving a major problem in modern power-hungry computational learning algorithms. While the hardware platform discussed here, self-learning electronic circuits, does indeed improve power efficiency, our work here focuses on how to achieve power efficiency in the learning process itself. As a result, our power-efficient learning approach may be easily adaptable to other neuromorphic hardware systems that can perform self-learning to offer *additional* power savings compared to only using efficient hardware.

ACKNOWLEDGMENTS

We thank Purba Chatterjee, Marc Z. Miskin, and Vijay Balasubramanian for insightful discussions and feedback. This research was supported by the U.S. Department of Energy, Office of Basic Energy Sciences, Division of Materials Sciences and Engineering Award No. DE-SC0020963 (M.S.), the National Science Foundation via the UPenn Nos. MRSEC/DMR-1720530 and MRSEC/DMR-DMR-2309043 (S.D. and D.J.D.), and DMR-2005749 (A.J.L.), and the Simons Foundation (No. 327939 to A.J.L.). D.J.D. and A.J.L. thank CCB at the Flatiron Institute as well as the Isaac Newton Institute for Mathematical Sciences under the program "New Statistical Physics in Living Matter" (EPSRC Grant No. EP/R014601/1) for support and hospitality while a portion of this research was carried out.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Menachem Stern: Conceptualization (lead); Data curation (lead); Formal analysis (lead); Methodology (equal); Software (lead); Writing – original draft (lead); Writing – review & editing (lead). **Sam Dillavou**: Conceptualization (supporting); Data curation (supporting); Formal analysis (supporting); Methodology (equal); Validation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Dinesh Jayaraman**: Conceptualization (supporting); Data curation (supporting); Formal analysis (supporting); Methodology (supporting); Writing – original draft (supporting); Writing – review & editing (supporting). **Douglas J. Durian**: Conceptualization (supporting); Funding acquisition (equal); Project administration (supporting); Resources (equal); Supervision (supporting); Writing – original draft (supporting); Writing – review & editing (supporting). **Andrea J. Liu**: Conceptualization (supporting); Funding acquisition (lead); Project administration (equal); Resources (equal); Supervision (lead); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are openly available in Stern, Menachem (2024). Data and codes for producing results associated with the manuscript "Training self-learning circuits for power-efficient solutions" are available at https://doi.org/10.6084/m9.figshare.24923685.v1.[51]

APPENDIX A: LEARNING DYNAMICS

Here, we provide more details on the derivation of the learning dynamics as well as how the free power P^F required to actuate the network response changes during learning. Before tackling the question of the free power of a learning network, let us study the dynamics of the learning DOF, k, and the contrast, $\mathcal{C}$, due to the learning rule in Eq. (2). We assume that there exists a solution of the learning degrees of freedom k^* such that the contrast vanishes $\mathcal{C}(k^*) = 0$ (this is the statement that the learning model is over-parameterized so that the learning degrees of freedom can be trained to nullify the training error). Over-parameterization implies the existence of many connected solutions in k space for which the contrast vanishes, and we denote by k^* the solution obtained in practice by learning. The contrast $\mathcal{C}$ is a complicated non-convex function of the learning DOF, but we can expand it around the solution k^* to first non-vanishing order (second order),

$$\mathcal{C}(k) \approx \frac{1}{2}(k-k^*)^T \mathcal{H}(k-k^*),$$

where $\mathcal{H} \equiv \partial_k^2 \mathcal{C}(k^*)$ is the "learning Hessian," i.e., the Hessian of the contrast with respect to the learning DOF evaluated at the solution. Close enough to the learning solution k^*, we find that despite the explicit partial differentiation in Eq. (2), the learning dynamics are equivalent to the gradient descent on the contrast.[34] Therefore, if we absorb the learning rate into the definition of the time unit, the weight dynamics are given by $\dot{k} = -\nabla_k \mathcal{C} = -\mathcal{H}(k-k^*)$. This leads to simple exponential decaying dynamics. If we set the initial condition at $k(t=0) \equiv k^0$, then

$$k(t) = k^* + e^{-\mathcal{H}t}(k^0 - k^*). \tag{A1}$$

Setting the time propagator operator $U(t) \equiv e^{-\mathcal{H}t} = U^T$, we can use this result to obtain the decaying dynamics of the contrast,

$$\mathcal{C}(t) = \frac{1}{2}(k^0 - k^*)^T U \mathcal{H} U (k^0 - k^*). \tag{A2}$$

While these results are consistent with simple exponential decay of the learning DOF k and contrast $\mathcal{C}$, one complication typically arises for over-parameterized learning. We have seen before that the learning Hessian in over-parameterized learning machines tends to be low rank (with the number of non-zero eigenvalues equal to the number of training tasks).[43] As the learning Hessian has zero eigenvalues, it is not invertible. There are no dynamics in the eigendirections of these vanishing eigenvalues, as can be explicitly seen by rotating the frame into the coordinate system that diagonalizes $\mathcal{H}$. The learning dynamics are agnostic to the components of k in the large null-space of $\mathcal{H}$. We can plug these results in Eq. (3) to obtain the free state power dynamics,

$$\dot{P}^F(k) = (k^* - k^0)^T U \mathcal{H} \partial_k P^F(k^*). \tag{A3}$$

As only $U(t)$ depends on time, this ODE can be integrated to find that the free power exponentially saturates to a value

$$P^F(t \to \infty) = P^F(t=0) + (k^* - k^0)^T A^T A \partial_k P^F(k^*), \tag{A4}$$

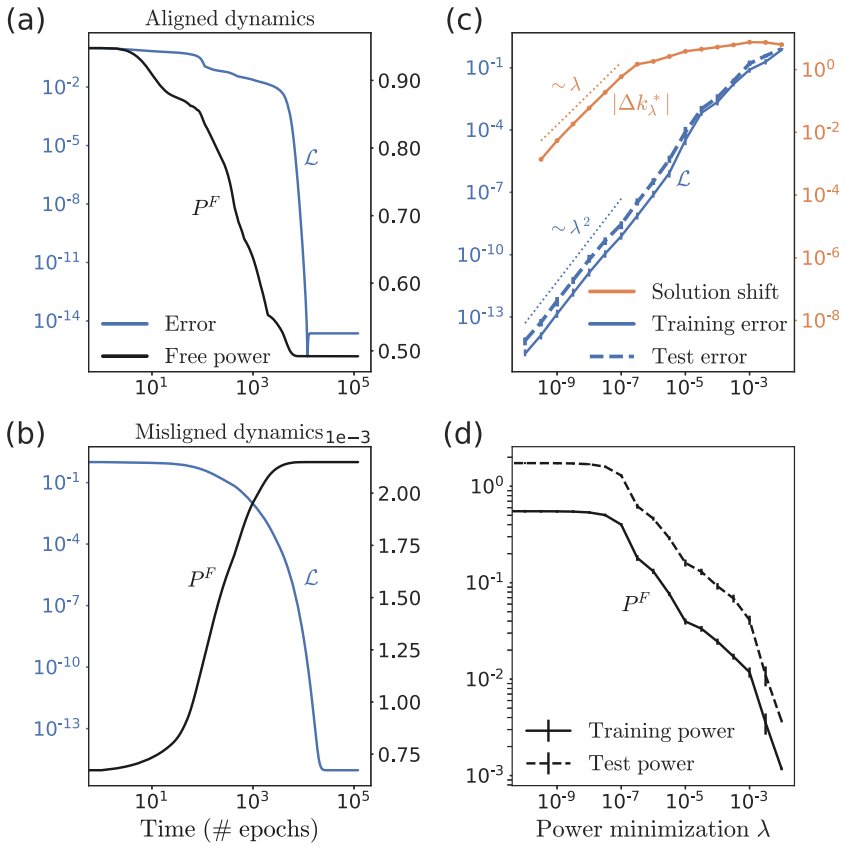

FIG. 5. Learning dynamics with power minimization. (a) Error $\mathcal{L}$ (blue) and free state power (black) as functions of time for a case where the gradients of the error and the free power align. We see that both are reduced by learning, and the error undershoots its steady state value before relaxing back to it. (b) Error $\mathcal{L}$ (blue) and free state power (black) as functions of time for a case where the gradients of the error and free power *do not* align. Here, learning increases the free power while smoothly reducing the error to its final value. (c) Solution shifts (orange) $|\Delta k_\lambda^*|$ as well as training error (full blue line) and test error (dashed blue line) as a function of the power minimization amplitude λ. As λ is increased, the learned solution k_λ^* linearly displaces from the limiting solution $k_{0^+}^*$. The error increases quadratically with λ, both for the training set and for the test set. (d) Trained free power for the training and test sets in the learned regression problem, both decrease as a function of λ. The results in panels (c) and (d) averaged over 50 realization of networks and tasks.

where A is a projection matrix, projecting weight vectors into the stiff (i.e., non-null) subspace of $\mathcal{H}$. Here, we see again that the power can increase or decrease during learning, depending on the alignment between the gradient of the free state power and the direction of weight dynamics.

We now discuss the modified learning dynamics that minimizes both error and free power [Eq. (9)]. In the main text, we showed that these learning dynamics lead to exponentially decaying weight solutions [Eq. (12)] and associated error and free power dynamics given in Eq. (15). The dynamical trajectories given these error\power dynamics follow two different prototypes, depending on the sign of $\phi \equiv -\partial_k \mathcal{C}^T \partial_k P^F$. For $\phi < 0$ (where the contrast gradient is aligned with the free power gradient), the contrast undershoots the infinite time limit, getting arbitrarily close to $\mathcal{C} = 0$ before rebounding exponentially to $\mathcal{C}(t \to \infty)$ [Fig. 5(a)]. This scenario is common when initializing the network with high conductance values. For $\phi > 0$ (i.e., anti-alignment of the contrast and free power gradients), the dynamics tend to increase the free power, and we see analytically regular dynamics, where the contrast smoothly decays exponentially to its terminal value $\mathcal{C}(t \to \infty)$ [Fig. 5(b)]. This scenario is common in flow\resistor networks initialized at low conductance values. Figure 5(c) shows the verification of the argument laid out in the main text that the solution $k^*_\lambda - k^*_{0^+} \sim \lambda$. We also show as before that the error grows quadratically with λ. Crucially, the arguments for the error are relevant not only for the training set (regression examples used to train the network) but also for the test examples that the network had not seen previously, whose error also scales quadratically in λ. More information about the regression tasks, as well as the training and test sets, is presented in Appendix B. Similarly to the error, the arguments about the trained free power are valid for both the training and test sets so that our modified learning dynamics reduces both of them [Fig. 5(d)].

It is also interesting to combine our results for the dependence of the free power on both the power optimization λ and the initialization scale of the learning DOF κ, as discussed in Sec. II B. We simulated the learning of regression tasks on $N = 64$ networks as earlier and varied λ in the range 10^{-10}–10^{-2} and the initialization scale κ in the allowed range 10^{-3}–10^{1}. We observe the emergence of two regimes of interest, one where the power minimization is weak $\lambda \ll 1$ and the other where λ is large [Fig. 6(a)]. The large λ regime is simpler to understand as the learning solutions there primarily reduce free power at the expense of error. Therefore, the free power reduction ΔP^F_λ is essentially the reduction from the free power at initialization to the minimal free power supported by the system, which scales as κ [at $\lambda = 10^{-4}$, Fig. 6(b)]. For weak minimization $\lambda \ll 1$, the effect of the initialization is more subtle. We find that the free power reduction scales approximately as a power law $\kappa^{0.5}$ [at $\lambda = 10^{-8}$, Fig. 6(b)]. Since here we measure the free power reduction, we find that at lower initialization scales, less power is saved by applying power minimization. However, there is still a substantial benefit in free power reduction even for good initialization. These results also help contextualize our dynamical control scheme in Sec. IV, where we show that the dynamical scheme supports additional free power saving compared to just using good initialization. We reserve the detailed study of the interplay between initialization and free power minimization for future study.

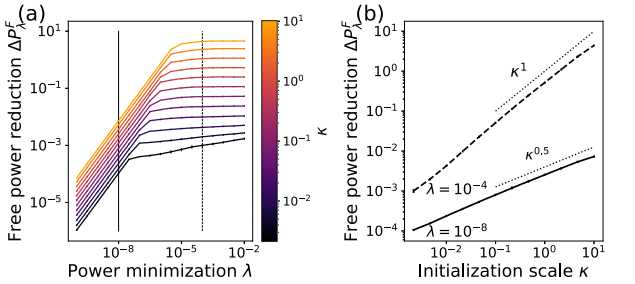

FIG. 6. Interplay of initialization and power minimization. (a) Free power reduction ΔP^F_λ depends on both the power minimization amplitude λ and the initialization scale κ. When using better initialization (lower κ), less power is saved by the power-efficient learning rule. (b) The free power ΔP^F_λ as a function of the initialization scale κ. For weak minimization ($\lambda = 10^{-8}$, full line), $\Delta P^F_\lambda \sim \kappa^{0.5}$. For strong power minimization ($\lambda = 10^{-4}$, dashed line), $\Delta P^F_\lambda \sim \kappa$. The results are averaged over ten realizations of networks and tasks.

APPENDIX B: PHYSICAL LEARNING TASKS

Here, we describe the regression tasks explored numerically in the main text. We simulated linear resistor networks with $N = 64$ whose structure is derived from jammed two-dimensional packings.[52] We randomly choose two edges as input edges and another two as output edges [see Fig. 1(a)]. The input and output voltage drops are noted by the vectors $\Delta V_i, \Delta V_o$, respectively. The network is trained to perform regression recovering a linear relation,

$$\Delta V_o + \epsilon = \sum_i \tilde{A}_{oi} \Delta V_i. \tag{B1}$$

Here, the 2×2 matrix $\tilde{A}_{oi}$ contains the desired function parameters and ϵ, a possible addition of white noise. Since we train a linear resistor network, the functional relation between the input and output voltage drops is always linear $\Delta V_o = \sum_i A_{oi} \Delta V_i$, and the correct matrix relation $\tilde{A}_{oi}$ is supposed to be recovered by learning. The values for the desired matrix were randomly chosen from the distribution

$$\tilde{A} \sim \begin{pmatrix} 0.2 & 0.3 \\ 0.1 & 0.5 \end{pmatrix} + 0.1 \mathcal{N}(0,1)^{2\times 2}. \tag{B2}$$

We trained these networks in many realizations of geometry, choice of input/output edges and $\tilde{A}_{oi}$. To train each realization of the problem, we sampled $n_{tr} = 20$ training examples $\Delta V_i^{\text{Training}} \sim U(0,1)^2$ and corresponding outputs $\Delta V_o^{\text{Training}} = \sum_i \tilde{A}_{oi} \Delta V_i^{\text{Training}} + \epsilon$. Note that the scale of the input voltage drops determines the scale of power dissipation in the free state $P^F \sim \Delta \tilde{V}_i{}^2$.

In the main text, we looked at noiseless regression problems with $\epsilon = 0$, for which, the network can find exact solutions with zero error. In Appendix C, we study a case with finite label noise $\epsilon = 10^{-3}$. The training examples are sampled randomly during training and used to define the free and clamped states in the iterative learning process. Apart from the $n_{tr} = 20$ training examples, we also sampled $n_{te} = 100$ test examples from a wider distribution $V_i^{\text{Test}} \sim \mathcal{N}(0,1)$ and their associated desired outputs. The test points are not used during the learning process but help in verifying that the network can generalize. In our work, the test set is also interesting for showing

the power-efficient property of the solutions that generalizes beyond the training set [Fig. 5(d)].

Once the training set is established, we can measure the error by simulating the output response of the network given input values, $\Delta V_O(\Delta V_i^{\text{Training}})$ and comparing it to the sample desired output values $\Delta V_O^{\text{Training}}$ by using a mean squared error loss function,

$$\mathcal{L} = 0.5 n_{tr}^{-1} \sum [\Delta V_O(\Delta V_i^{\text{Training}}) - \Delta V_O^{\text{Training}}]^2.$$

Similarly, we can compute the loss value associated with the test set. We simulate physical learning by applying the local learning rules described in the main text, picking a nudging amplitude value $\eta = 10^{-3}$.

For better comparison to the physical experiment of Fig. 3, we also simulated a resistor network of the same geometry and a simple allostery task as the experiment [inset of Fig. 3(d)]. In these simulations, we randomly choose two input nodes and assign to them input voltages, $V_{i1} = 0V$ and $V_{i2} = 0.45V$. We further choose two random output nodes and train them such that when the inputs are applied, they would have output voltage values $V_{o1} = 0.09V$ and $V_{o2} = 0.36V$. In this case, we again use a mean squared error loss function and compare the output voltages at the output nodes to the desired values $V_{o1,2}$.

APPENDIX C: POWER MINIMIZATION FOR LIMITED ACCURACY TASKS

The numerical results in the main text were limited to tasks that can, in principle, be learned perfectly by the learning machine. In such cases, there exist solutions with no error $\mathcal{L}(k^*) = 0$, as discussed in Appendix A. There are, however, cases in which it is impossible to obtain solutions with zero error. The typical example is when the training set does not capture all the information contained in the broader data (or the test set). There are also cases where it is impossible to find solutions that nullify the error even on the training set. This can occur due to under-parameterization (too few learning degrees of freedom to learn the task), an insufficiently expressive model (e.g., a linear network cannot represent nonlinear relations) and noise in the learning process.[48]

First, we will consider the case where the system ends up in a local minimum with $\mathcal{L} > 0$. From the definition of coupled learning, we know that if the loss is finite $\mathcal{L} > 0$ so is the contrast $\mathcal{C} > 0$. In such as case, a minimum of the coupled learning dynamics $k^\dagger$ would have finite error and contrast values $\mathcal{L}(k^\dagger)$, $\mathcal{C}(k^\dagger)$. Nonetheless, we can still perform a quadratic approximation around the contrast minimum $k^\dagger$ similar to Appendix A, where the constant term $\mathcal{C}(k^\dagger)$ is retained,

$$\mathcal{C}(k) \approx \mathcal{C}(k^\dagger) + \frac{1}{2}(k - k^\dagger)^T \mathcal{H}(k - k^\dagger).$$

Using this expansion, we can redo the derivation in Sec. III to find the steady state solution error and the trained free power when a finite power minimization amplitude λ is applied in Eq. (9),

$$\begin{aligned} \mathcal{C}_\lambda &\approx \mathcal{C}(k^\dagger) + \frac{1}{2}\lambda^2 s^T \mathcal{H} s, \\ \Delta P_\lambda^F &\approx \lambda(\partial_k P_{0^+}^F)^T(\mathcal{H} + \lambda H)^{-1}\partial_k P_{0^+}^F. \end{aligned} \tag{C1}$$

Comparing these expressions to Eq. (16), we see that the trained free power behavior stays the same. We also see that the error shift is the same, scaling as λ^2, but now there is a finite contrast floor $\mathcal{C}(k^\dagger)$ associated with a finite error. The trade-off between error and power is still maintained, although in this case, it may be much more favorable. For a small enough λ, $\frac{1}{2}\lambda^2 s^T \mathcal{H} s \ll \mathcal{C}(k^\dagger)$, and so the contrast (and error) is nearly unaffected by the power minimization. As a result, we can apply a finite power minimization parameter λ, reducing the trained free power at nearly no penalty. Thus, power minimization is particularly useful for problems in which zero error solutions are not possible.

To verify these considerations, we simulated physical learning in $N = 64$ networks on regression and classification tasks (Fig. 7). Excess noise was added to the regression labels (outputs) in the training and test sets, sampled from a distribution $\Delta V_o \sim \sum_i \tilde{A}_{oi}\Delta V_i + \epsilon$, with $\epsilon = 10^{-3}$ (see Appendix B). The simulated networks can successfully learn these tasks, reducing the error to some finite value $\mathcal{L} \approx 10^{-5}$ [Fig. 7(a)]. When adding a small power minimization $\lambda < 10^{-7}$, the learning trajectories are almost unchanged and the error is nearly unaffected. When λ increases, the error starts increasing beyond the error floor [Fig. 7(b)]. At the same time, we observe that the trained free power is decreased linearly at finite λ, as seen before. These results show that in noisy cases, such as those seen in physical learning experiments, free power reduction can be achieved at little expense in errors up to a certain point.

Another case where this result is particularly relevant is in classification problems, where we would like to assign discrete labels to inputs. A standard example for such tasks is the classification of Iris specimens based on the measurements of lengths of their petals and sepals.[53] Previously, we have shown that our flow\resistor networks can successfully learn to classify the Iris dataset, as well as could be expected from linear network models, in simulations[43] and experiments.[37] In discrete classification tasks, we are typically not concerned with the mean squared error but with a measure of accuracy given by a discrete choice of the label based on the network response; excellent classification is possible even at relatively high values of the mean squared error. Therefore, it may be possible to induce power optimization without a penalty in classification accuracy. To test this idea, we simulated the training of our $N = 64$ node networks to classify the Iris dataset (a detailed description of the training protocol can be found in Ref. 37). Training at different power minimization amplitudes λ in the range $10^{-10} < \lambda < 10^{-2}$, we find that the classification accuracy (for the training and test sets) is not affected by power minimization until $\lambda \approx 10^{-7}$ [Fig. 7(c)]. At the same time, the solution free power is significantly reduced starting at $\lambda > 10^{-8}$, showing that power gain (in this case, by a factor ~2) is possible at little penalty in accuracy [Fig. 7(d)].

Now, we turn to another case in which tasks cannot be learned perfectly, this time due to the existence of noise. In any real physical learning, machine noise in measurement and learning DOF updates will lead to a non-zero error floor associated with physical learning. This is true even for tasks that, in the absence of noise, could be learned with no error. In such a setting, the random noise pushing the system away from the zero contrast (and error) minima implies that physical learning behaves as a high dimensional Ornstein–Uhlenbeck process[54] in the space of the learning DOF. The instantaneous values of the learning DOF are then sampled from a

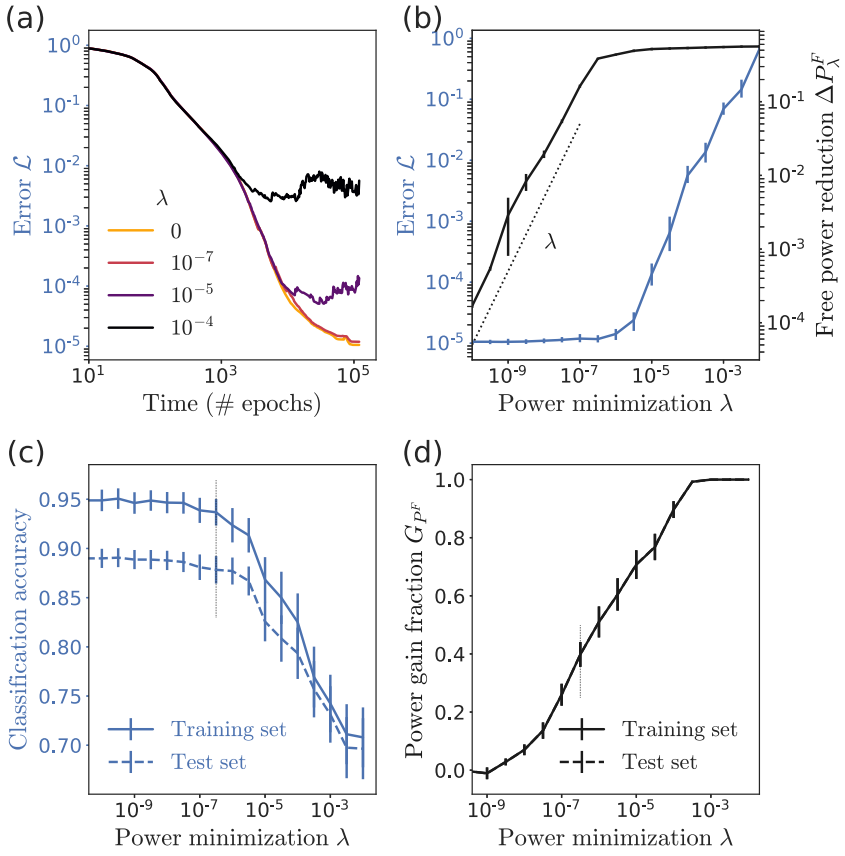

FIG. 7. Power reduction with little error\accuracy loss in regression and classification problems. (a) Error vs time trajectories for regression task with label noise (with minimum possible error $\mathcal{L} \approx 10^{-5}$) for different values of the power minimization amplitude λ. As long as $\lambda < 10^{-7}$, the error is largely unaffected. (b) Error (blue) and free power reduction (black) as a function of λ. The trained free power is still reduced by increasing λ, as before, even in the range where the error is unaffected. The regression results averaged over 50 realizations. (c) Classification accuracy of the Iris dataset for the training (full line) and test (dashed line) sets, as a function of λ. Similar results are obtained, as increasing the power minimization amplitude λ decreases accuracy (i.e., increases error), but only beyond a finite value of λ. (d) Power gain ΔG_{P^F} due to power minimization for the training (full line) and test (dashed line) sets. In this case, we gain a factor 2 in trained free power with little loss in accuracy (at $\lambda \approx 10^{-7}$). The classification results averaged over 100 realizations.

normal distribution centered around k^*_λ in Eq. (14), with a standard deviation scaling with the white noise amplitude σ,[55]

$$k_{\lambda,i}(\sigma) \sim k^*_{\lambda,i}(\sigma = 0) + \frac{\sigma}{\sqrt{2\theta_{\lambda,i}}}\mathcal{N}(0,1), \tag{C2}$$

where $\theta_{\lambda,i}$ are the eigenvalues of the matrix $\mathcal{H} + \lambda H$. In other words, the noise induces the conductances to explore a vicinity of the solution k^*_λ, whose size depends on the noise amplitude σ, and the curvature is given by the eigenvalues $\theta_{\lambda,i}$. We can take this distribution of values of the learning DOF and plug it in the equation for the contrast [Eq. (11)], finding the distributions of this quantity,

$$\mathcal{C}_\lambda(\sigma) \sim \frac{1}{2}\lambda^2 s^T \mathcal{H} s + \sum_i \frac{\lambda\sigma s_i}{\sqrt{2\theta_{\lambda,i}}}\mathcal{N}_i(0,1) + \sum_i \frac{\sigma^2}{4\theta_{\lambda,i}}\mathcal{N}_i^2(0,1). \tag{C3}$$

Similarly, this can be done for the free power reduction ΔP^F_λ. The average contrast induced by the noise, as well as the average free power reduction, can be deduced by taking the expectation value over these distributions. Here, note that the expectation values of these normal distributions are $\langle \mathcal{N}_i(0,1)\rangle = 0, \langle \mathcal{N}_i^2(0,1)\rangle = 1$ so that we are left with

$$\begin{aligned} \langle \mathcal{C}_\lambda(\sigma)\rangle &\approx \frac{1}{2}\lambda^2 s^T \mathcal{H} s + \sum_i \frac{\sigma^2}{4\theta_{\lambda,i}}, \\ \langle \Delta P^F_\lambda\rangle(\sigma) &\approx \lambda(\partial_k P^F_{0^+})^T(\mathcal{H} + \lambda H)^{-1}\partial_k P^F_{0^+}. \end{aligned} \tag{C4}$$

We find that if we know the noise scale σ, measuring the average contrast value $\langle \mathcal{C}_0(\sigma)\rangle$ allows us to glean information about the effective average curvature of the contrast near the learning solution. Note that the learning DOF diffuse freely in the space of zero contrast solutions, so the effective curvature is associated with the typical slopes of the contrast leaving the zero manifold. Overall, we see that the free power reduction is, on average, the same as in the case with no noise (up to second order terms in λ). However, the contrast now has a finite added term due to the exploration of values of the

learning DOF beyond the minimum k_λ^*. This means that additive white noise has a similar effect to the finite contrast floor discussed earlier; finite power minimization λ can reduce the trained free power while having nearly no effect on the contrast (or error) up to a certain scale.

APPENDIX D: POWER MINIMIZATION IN MECHANICAL SPRING NETWORKS

In this work, we presented general arguments on how local learning rules could balance minimizing the error and trained-free power of obtained physical learning solutions, giving rise to a trade-off between the two. However, in the main text, we only tested these ideas numerically and experimentally in resistor networks. Here, we show in simulations that these arguments apply similarly to physical learning systems governed by different physics, e.g., an elastic network of harmonic springs [Fig. 8(a)].

Elastic networks have been studied as a nonlinear substrate for physical learning.[49,50,56–60] Specifically, coupled learning can train spring networks to perform the desired tasks by modifying the spring constants or rest lengths.[34] The physical cost function naturally minimized by such networks is the elastic energy E,

$$E = \frac{1}{2}\sum_i k_i (r_i - \ell_i)^2, \tag{D1}$$

where k_i is the spring constant of spring i, ℓ_i is its rest length, r_i the Euclidean distance between the nodes connected by the spring, and the energy is summed over all individual springs. For a spring network with adaptive spring constants, the local learning rule is

$$\begin{aligned}\dot{k}_i &= -\alpha\eta^{-1}\frac{\partial}{\partial k_i}[E^C - E^F] \\ &= -\frac{1}{2}\alpha\eta^{-1}[(r_i^C - \ell_i)^2 - (r_i^F - \ell_i)^2],\end{aligned} \tag{D2}$$

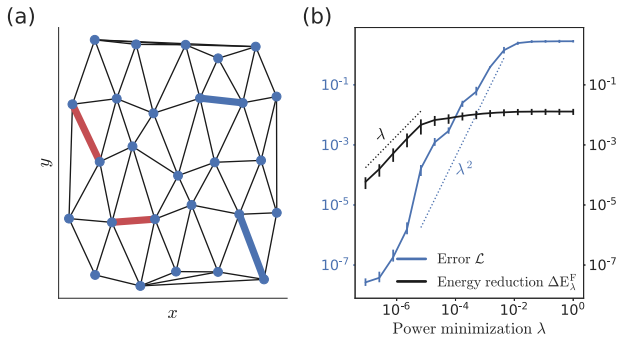

FIG. 8. Energy-efficient learning in mechanical spring networks. (a) A mechanical spring network, each edge corresponding to a spring with adaptive stiffness k. Such networks are trained for allostery tasks so that prescribed strains at input edges (red) lead to desired strains at output edges (blue). (b) Error $\mathcal{L}$ (blue) and free energy reduction ΔE_λ^F (black) as functions of the power minimization amplitude λ. As seen for flow networks, including a power minimization term in the local learning rule leads to a trade-off between error and trained free energy, also having the same scaling behaviors. The results are averaged over five realizations of networks and tasks.

where r_i^F, r_i^C are the distances between the nodes separated by spring i in the free and clamped state, respectively. More details on the derivation of this learning rule can be found in Ref. 34. To see if spring networks can be trained to adopt low free energy solutions, i.e., spring configurations for which the desired state is easy (takes little energy) to actuate, we add a local free energy minimization term with amplitude λ, similarly to Eq. (9),

$$\dot{k}_i = -\alpha\eta^{-1}\frac{\partial}{\partial k_i}[E^C - (1-\lambda)E^F]. \tag{D3}$$

We simulate this modified learning algorithm on an unstrained spring network with $N = 27$ nodes, as shown in Fig. 8(a). These networks are trained for allostery tasks, in which we apply prescribed relative strains 0.2 (randomly choosing contraction or extension) and desire particular strain values at another two random bonds (0.05 or 0.03, randomly choosing contraction or extension). With no energy minimization applied, $\lambda = 0$, and coupled learning generally succeeds in training these networks to a numerical normalized error floor of $\mathcal{L} \sim 10^{-8}$. As we increase the power minimization amplitude λ, we observe that the error increases as λ^2 and the trained free energy E^F is reduced as λ [Fig. 8(a)], as predicted by Eq. (16) and observed in simulations of resistor networks. These results show that our approach to physical learning of power efficient solutions can be employed beyond linear resistor networks. Recent experimental progress has been achieved for implementing coupled learning in elastic networks,[42] but we leave the experimental validation of energy reduction in such networks for future study.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**(7553), 436–444 (2015).
[2] P. P Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)* (IEEE, 2018), pp. 1–6.
[3] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *2017 IEEE Custom Integrated Circuits Conference (CICC)* (IEEE, 2017), pp. 1–8.
[4] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," J. Parallel Distrib. Comput. **134**, 75–88 (2019).
[5] D. Amodei, D. Hernandez, G. Sastry, J. Clark, G. Brockman, and I. Sutskever, Ai and Compute, 2018; https://openai.com/research/ai-and-compute Accessed 16 June 2023.
[6] A. Van Wynsberghe, "Sustainable AI: Ai for sustainability and the sustainability of AI," AI Ethics **1**(3), 213–218 (2021).
[7] U. Gupta, G. YoungKim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Chasing carbon: The elusive environmental footprint of computing," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (IEEE, 2021), pp. 854–867.
[8] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai *et al.*, "Sustainable AI: Environmental implications, challenges and opportunities," Proc. Mach. Learn. Syst. **4**, 795–813 (2022).
[9] C. Mead, "Neuromorphic electronic systems," Proc. IEEE **78**(10), 1629–1636 (1990).
[10] G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola *et al.*, "Neuromorphic computing using non-volatile memory," Adv. Phys.: X **2**(1), 89–124 (2017).
[11] D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, "Physics for neuromorphic computing," Nat. Rev. Phys. **2**(9), 499–510 (2020).
[12] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," Nat. Comput. Sci. **2**(1), 10–19 (2022).

[13]D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. Le Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel *et al.*, "2022 roadmap on neuromorphic computing and engineering," Neuromorphic Comput. Eng. **2**(2), 022501 (2022).
[14]J. Hasler and B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," Front. Neurosci. **7**, 118 (2013).
[15]M. Sharad, C. Augustine, G. Panagopoulos, and K. Roy, "Proposal for neuromorphic hardware using spin devices," arXiv:1206.3227 (2012).
[16]K. IvanSchuller, R. Stevens, R. Pino, and M. Pechan, "Neuromorphic computing–from materials research to systems architecture roundtable," Technical Report [USDOE Office of Science (SC), USA, 2015].
[17]M. Kang, Y. Lee, and M. Park, "Energy efficiency of machine learning in embedded systems using neuromorphic hardware," Electronics **9**(7), 1069 (2020).
[18]M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud, "Advancing neuromorphic computing with loihi: A survey of results and outlook," Proc. IEEE **109**(5), 911–934 (2021).
[19]E. O. Neftci, "Data and power efficient intelligence with neuromorphic learning machines," Iscience **5**, 52–68 (2018).
[20]M. Sorbaro, Q. Liu, M. Bortone, and S. Sheik, "Optimizing the energy consumption of spiking neural networks for neuromorphic applications," Front. Neurosci. **14**, 662 (2020).
[21]I. Chakraborty, A. Jaiswal, A. K. Saha, S. K. Gupta, and K. Roy, "Pathways to efficient neuromorphic computing with non-volatile memory technologies," Appl. Phys. Rev. **7**(2), 021308 (2020).
[22]N. Simon, S. A. Bigdeli, S.-C. Liu, and L. A. Dunbar, "Optimizing the consumption of spiking neural networks with activity regularization," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2022), pp. 61–65.
[23]M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)* (IEEE, 2016), pp. 1–6.
[24]Z. Wang, C. Li, W. Song, M. Rao, D. Belkin, Y. Li, P. Yan, H. Jiang, P. Lin, M. Hu, J. P. Strachan, N. Ge, M. Barnell, Q. Wu, A. G. Barto, Q. Qiu, R. S. Williams, Q. Xia, and J. J. Yang, "Reinforcement learning with analogue memristor arrays," Nat. Electron. **2**(3), 115–124 (2019).
[25]W. Zhang, B. Gao, J. Tang, P. Yao, S. Yu, M.-F. Chang, H.-J. Yoo, H. Qian, and H. Wu, "Neuro-inspired computing chips," Nat. Electron. **3**(7), 371–382 (2020).
[26]Y. Arima, M. Murasaki, T. Yamada, A. Maeda, and H. Shinohara, "A refreshable analog VLSI neural network chip with 400 neurons and 40 K synapses," IEEE J. Solid-State Circuits **27**(12), 1854–1861 (1992).
[27]C. R. Schneider and H. C. Card, "Analog CMOS deterministic Boltzmann circuits," IEEE J. Solid-State Circuits **28**(8), 907–914 (1993).
[28]S. Kim, C. Du, P. Sheridan, W. Ma, S. H. Choi, and W. D. Lu, "Experimental demonstration of a second-order memristor and its ability to biorealistically implement synaptic plasticity," Nano Lett. **15**(3), 2203–2211 (2015).
[29]S. La Barbera, A. F. Vincent, D. Vuillaume, D. Querlioz, and F. Alibart, "Interplay of multiple synaptic plasticity features in filamentary memristive devices for neuromorphic computing," Sci. Rep. **6**(1), 39216 (2016).
[30]A. Serb, J. Bill, A. Khiat, R. Berdan, R. Legenstein, and T. Prodromakis, "Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses," Nat. Commun. **7**(1), 12611 (2016).
[31]M. Stern and A. Murugan, "Learning without neurons in physical systems," Annu. Rev. Condens. Matter Phys. **14**(1), 417–441 (2023).
[32]J. R. Movellan, "Contrastive Hebbian learning in the continuous Hopfield model," in *Connectionist Models* (Elsevier, 1991), pp. 10–17.
[33]B. Scellier and Y. Bengio, "Equilibrium propagation: Bridging the gap between energy-based models and backpropagation," Front. Comput. Neurosci. **11**, 24 (2017).
[34]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**(2), 021045 (2021).
[35]V. R. Anisetti, B. Scellier, and J. M. Schwarz, "Learning by non-interfering feedback chemical signaling in physical networks," Phys. Rev. Res. **5**(2), 023024 (2023).
[36]V. Rao Anisetti, A. Kandala, B. Scellier, and J. M. Schwarz, "Frequency propagation: Multi-mechanism learning in nonlinear physical networks," arXiv:2208.08862 (2022).
[37]S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian, "Demonstration of decentralized physics-driven learning," Phys. Rev. Appl. **18**(1), 014040 (2022).
[38]J. F. Wycoff, S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian, "Desynchronous learning in a physics-driven learning network," J. Chem. Phys. **156**(14), 144903 (2022).
[39]M. Stern, S. Dillavou, M. Z. Miskin, D. J. Durian, and A. J. Liu, "Physical learning beyond the quasistatic limit," Phys. Rev. Res. **4**(2), L022037 (2022).
[40]S. Dillavou, B. Beyer, M. Stern, M. Z. Miskin, A. J. Liu, and D. J. Durian, "Circuits that train themselves: Decentralized, physics-driven learning," Proc. SPIE **12438**, 115–117 (2023).
[41]S. Dillavou, B. D. Beyer, M. Stern, M. Z. Miskin, A. J. Liu, and D. J. Durian, "Machine learning without a processor: Emergent learning in a nonlinear electronic metamaterial," arXiv:2311.00537 (2023).
[42]L. E. Altman, M. Stern, A. J. Liu, and D. J. Durian, "Experimental demonstration of coupled learning in elastic networks," arXiv:2311.00170 (2023).
[43]M. Stern, A. J. Liu, and V. Balasubramanian, "The physical effects of learning," arXiv:2306.12928 (2023).
[44]M. Fernández-Redondo and C. Hernández-Espinosa, "Weight initialization methods for multilayer feedforward," in ESANN, 2001, pp. 119–124.
[45]I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International Conference on Machine Learning* (PMLR, 2013), pp. 1139–1147.
[46]Y. Bahri, J. Kadmon, J. Pennington, S. S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli, "Statistical mechanics of deep learning," Annu. Rev. Condens. Matter Phys. **11**(1), 501 (2020).
[47]M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural networks," Artif. Intell. Rev. **55**(1), 291–322 (2022).
[48]P. Mehta, M. Bukov, C.-H. Wang, A. G. R. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists," Phys. Rep. **810**, 1 (2019).
[49]N. Pashine, D. Hexner, A. J. Liu, and S. R. Nagel, "Directed aging, memory, and nature's greed," Sci. Adv. **5**(12), eaax4215 (2019).
[50]D. Hexner, N. Pashine, A. J. Liu, and S. R. Nagel, "Effect of directed aging on nonlinear elasticity and memory formation in a material," Phys. Rev. Res. **2**(4), 043231 (2020).
[51]M. Stern, Data and Codes for producing results associated with the manuscript "Training self-learning Circuits for power-efficient solutions" (2024), p. 1.
[52]C. P. Goodrich, A. J. Liu, and S. R. Nagel, "The principle of independent bond-level response: Tuning by pruning to exploit disorder for global behavior," Phys. Rev. Lett. **114**(22), 225501 (2015).
[53]R. A. Fisher, "The use of multiple measurements in taxonomic problems," Ann. Eugen. **7**(2), 179–188 (1936).
[54]P. Vatiwutipong and N. Phewchean, "Alternative way to derive the distribution of the multivariate Ornstein–Uhlenbeck process," Adv. Differ. Equations **2019**(1), 276.
[55]C. W. Gardiner *et al.*, *Handbook of Stochastic Methods* (Springer Berlin, 1985), Vol. 3.
[56]D. Hexner, A. J. Liu, and S. R. Nagel, "Periodic training of creeping solids," Proc. Natl. Acad. Sci. U. S. A. **117**(50), 31690–31695 (2020).
[57]M. Stern, M. B. Pinson, and A. Murugan, "Continual learning of multiple memories in mechanical networks," Phys. Rev. X **10**(3), 031044 (2020).
[58]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. U. S. A. **117**(26), 14843–14850 (2020).
[59]D. Hexner, "Adaptable materials via retraining," arXiv:2103.08235 (2021).
[60]C. Arinze, M. Stern, S. R. Nagel, and A. Murugan, "Learning to self-fold at a bifurcation," Phys. Rev. E **107**(2), 025001 (2023).