Supervised Learning in Physical Networks: From Machine Learning to Learning Machines

Menachem Stern[ID],[1] Daniel Hexner,[2] Jason W. Rocks,[3] and Andrea J. Liu[ID][1]

[1]*Department of Physics and Astronomy, University of Pennsylvania,*
Philadelphia, Pennsylvania 19104, USA
[2]*Department of Mechanical Engineering, Technion, Haifa, 32000, Israel*
[3]*Department of Physics, Boston University, Boston, Massachusetts 02139, USA*

Materials and machines are often designed with particular goals in mind, so that they exhibit desired responses to given forces or constraints. Here we explore an alternative approach, namely physical *coupled learning*. In this paradigm, the system is not initially designed to accomplish a task, but physically adapts to applied forces to develop the ability to perform the task. Crucially, we require coupled learning to be facilitated by physically plausible learning rules, meaning that learning requires only local responses and no explicit information about the desired functionality. We show that such local learning rules can be derived for any physical network, whether in equilibrium or in steady state, with specific focus on two particular systems, namely disordered flow networks and elastic networks. By applying and adapting advances of statistical learning theory to the physical world, we demonstrate the plausibility of new classes of smart metamaterials capable of adapting to users' needs *in situ*.

Subject Areas: Metamaterials, Soft Matter,
Statistical Physics

I. INTRODUCTION

Engineered materials are typically designed to have particular sets of properties or functions [1]. The design process often involves numerous trial and error iterations, during which the system is repeatedly tested for the desired functionality [2], modified, and tested again. Alternatively, rational design processes start from detailed knowledge of material components and typically use computation to predict the consequences of tweaking the system to sift through many possibilities.

A second class of strategies is based on learning, where systems can adjust or be adjusted at the microscopic scale, in response to training examples, to develop the desired functionality. Until recently, learning strategies were primarily restricted to nonphysical networks such as neural networks on a computer. One class of methods, which we refer to as "global supervised learning," is ubiquitous for problems such as data classification [3,4]. These methods are based on the optimization of a cost or loss function that is "global" in that it depends on all of the microscopic details of the system. In the context of neural networks, for example, global supervised learning optimizes the network

by modifying a set of learning degrees of freedom (e.g., weights) controlling the signal propagation between the input and output sections of the network.

Global supervised learning was recently used to design flow and elastic networks—actual physical systems—with particular desired functions, e.g., allostery [5,6]. In this physical context, such learning methods were dubbed "tuning," since modifying the learning degrees of freedom (e.g., pipe conductances or spring constants in flow or mechanical networks, respectively) requires external intervention at the microscopic level.

In contrast, natural systems such as the brain evolve to obtain desired functions using a fundamentally different framework of learning, which we refer to as "local learning." Crucially, this evolution is entirely autonomous, requiring no external designer for evaluation of the current state of the system and its subsequent modification. In local learning, parts of the network can only adapt due to the local information available in their immediate vicinity (e.g., a synapse adapts in response to the activities only of the neurons directly connected to it [7]). It is particularly useful to apply this learning approach in physical networks such as flow or mechanical networks because the microscopic elements of such networks cannot perform computations and do not encode information about the desired functionality *a priori*.

Training physical networks for desired function, using either global or local supervised learning, involves two different sets of degrees of freedom. First, networks respond to source constraints by minimizing a scalar

function with respect to their physical degrees of freedom. For example, central-force spring networks minimize the elastic energy by adjusting the positions of their nodes to achieve force balance on every node, while flow networks minimize the power dissipation by adjusting the currents on edges to obey Kirchhoff's law at every node. Second, networks can learn specific desired target responses to sources by adjusting their learning degrees of freedom. These degrees of freedom could correspond to the stiffnesses or equilibrium lengths of springs in a mechanical network or the conductances of edges in a flow network. In global supervised learning, these degrees of freedom are adjusted to optimize the cost function.

We consider the case where the desired outcome of the learning process is to achieve tailored responses of "target" edges or nodes to external constraints applied at "source" edges or nodes. For example, an allosteric response in a mechanical network corresponds to a desired strain at a set of target edges in response to a strain applied at source edges. Similarly, for the simplest "flow allosteric" response, a pressure drop across a source edge in a flow network leads to a desired pressure drop across a target edge elsewhere in the network [8].

Recently, it was shown that a local supervised learning process of "directed aging," in which the time evolution of a disordered system is driven by applied stresses, can be used to create mechanical metamaterials with desired properties or functions [9–11]. For example, Refs. [10,11] consider a form of directed aging for a mechanical spring network in which the learning degrees of freedom are the equilibrium lengths of the springs comprising the network edges. The equilibrium length of each spring lengthens (shortens) if the spring is placed under extension (compression)—this is a local response to local extension (compression). Isotropic compression [9,10] or repeated cycles of isotropic compression and expansion can drive the Poisson's ratio of such a network from positive to negative values, while cycles of compression or stretching oscillations of source and target edges can lead to allosteric response [11].

Similarly, local rules in growing vascular networks [12] and folding sheets [13,14] allow those systems to learn properties or functions autonomously. The great advantage of local supervised learning such as directed aging over global supervised learning is that the process is scalable—it can be applied to train large systems without having to manually modify their parts [9]. In addition, directed aging does not require detailed knowledge of microscopic interactions or the ability to manipulate (microscopic) learning degrees of freedom [9].

While directed aging methods are successful in training certain physical networks for desired functions, they fail in others, particularly in highly overconstrained networks such as flow networks and high-coordination mechanical networks. Failure occurs because directed aging minimizes the physical cost function of a desired state of the network, rather than the cost function whose minimization corresponds directly to the desired function in global supervised learning.

Here we propose a general framework, which we call "coupled local supervised learning," for physical networks such as flow or mechanical networks. The rules are designed to adjust learning degrees of freedom in response to local conditions such as the tension on a spring in a mechanical network or the current through a pipe of a flow network. The framework provides a way of deriving rules that lead to modifications of learning degrees of freedom that are extremely similar to those obtained by minimizing the cost function. As a result, they are as likely to succeed in obtaining the desired response as global supervised learning would be.

The coupled learning framework is inspired by advances in neuroscience and computer science [15–19] known as "contrastive Hebbian learning." As in contrastive Hebbian learning, one considers the response in two steady states of the system, one in which only source constraints are applied (free state), the other where source and target constraints are applied simultaneously (clamped state). The particular rules we introduce, which we call "coupled learning rules," are also inspired by the strategy of "equilibrium propagation" [18], which promotes infinitesimal nudging and hence close proximity between the free and clamped states. In Sec. II, we first show in detail how coupled learning works for flow networks (Sec. II A). We successfully train such networks to obtain complex functionalities. We then discuss the general framework of coupled learning in generic physical networks (Sec. II B) and apply these ideas to nonlinear elastic networks (Sec. II C). We demonstrate our learning framework on a standard classification problem, distinguishing handwritten digits (Sec. II D).

It is important to note that implementation of coupled learning should be possible in real systems. In Sec. III, we therefore consider complications that may arise in realistic learning scenarios. We first derive an approximate version of the local learning rule that may be more easily implemented experimentally in a flow network (Sec. III A). While inferior to the full learning rule, approximate coupled learning still gives rise to desired functionality. We then discuss limitations due to noisy measurements of the physical degrees of freedom, that limit the usefulness of small nudging, and the implications of drifting in the learning degrees of freedom (Sec. III B). Finally, we address the effect of network size on the physical relaxation time and the coupled learning rules (Sec. III C).

Following the introduction of the coupled learning rules, in Sec. IV we compare coupled learning to standard global supervised learning frameworks that minimize a cost function [5,6,20,21], discussing the experimental realizability of coupled learning in contrast with such methods. We hope this work will stimulate further interest in
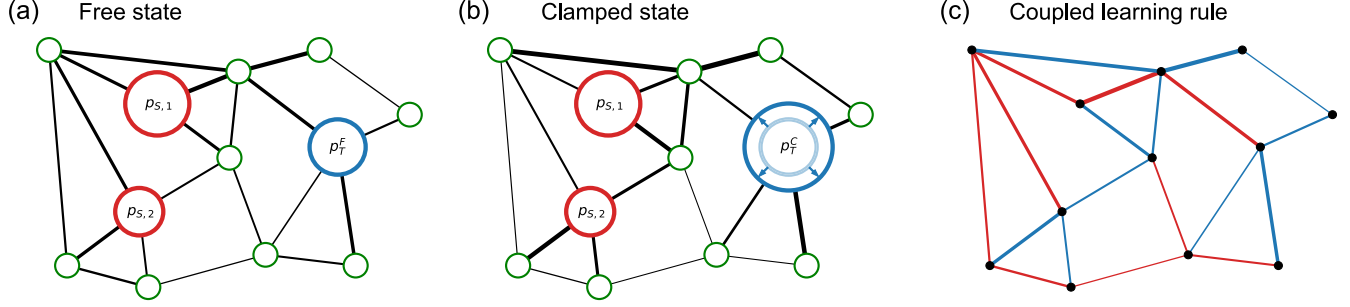
FIG. 1. Coupled learning in flow networks. (a) In the free phase, node pressures are constrained such that source nodes (red) have specific pressure values p_S. The target node pressures p_T and the dissipated power at all pipes $\mathcal{P}_j$ attain their steady-state values due to the natural flow processes in the network. (b) In the clamped phase, a supervisor constrains the target node pressures so that they are slightly closer to their desired values compared to the free state. (c) A strictly local learning rule, i.e., the modification of pipe conductance according to its response, is proportional to the difference in dissipated power between the free and clamped states. The learning rule thus couples the pressures at the source and target nodes.

physically inspired learning, opening possibilities for new classes of metamaterials and machines, able to autonomously evolve and adapt in desired ways due to external influences.

II. COUPLED LEARNING FRAMEWORK

A. Coupled learning in flow networks

We first discuss coupled learning in the context of flow networks. Previously, such networks were trained to exhibit the particular function of flow allostery [5,6] using global supervised learning by minimizing a global cost function. Specifically, the networks were trained to have a desired pressure drop across a target edge (or many target edges) in response to a pressure drop applied across a source edge elsewhere in the network. Here we show how a strictly local learning rule can similarly train flow networks.

A flow network is defined by a set of N nodes labeled μ, each carrying a pressure value p_μ; these pressures are the physical degrees of freedom of the network. The nodes are connected by pipes j characterized by their conductances k_j; these conductances are the learning degrees of freedom because modifying the pipe conductances will enable the network to develop the desired function. Assuming each pipe is directed from one node μ_1 to the other μ_2, the flow current in the pipe is given by $I_j = k_j(p_{\mu_1} - p_{\mu_2}) \equiv k_j \Delta p_j$. If boundary conditions are applied to the network (e.g., fixed pressure values at some nodes) the network finds a flow steady state, in which the total dissipated power,

$$\mathcal{P} = \frac{1}{2} \sum_j k_j \Delta p_j^2, \qquad (1)$$

is minimized by varying the pressures at unconstrained nodes.

We now define a task for the network to learn as follows. We subdivide the physical degrees of freedom (node pressures) $\{p_\mu\}$ into three types, corresponding to source

nodes p_S, target nodes p_T, and the remaining "hidden" nodes p_H. The desired task is defined such that the target node pressure values reach a set of desired values $\{P_T\}$ when the source node pressures are constrained to the values $\{P_S\}$. A generic disordered flow network does not possess this function, so design strategies are needed to identify appropriate values for the pipe conductance values $\{k_j\}$ that achieve the desired task.

For the network to learn, or adapt autonomously to source pressure constraints, we allow the learning degrees of freedom $\{k_j\}$ (pipe conductances) to vary depending on the physical state of the network $\{p_\mu\}$. A *learning rule* is an equation of motion for the learning degrees of freedom, taking the form:

$$\dot{k}_j = f(p_\mu; k_j). \qquad (2)$$

We focus on local learning rules, where the learning degree of freedom k_j in each pipe j can only change in response to the physical variables (pressure values) $\{p_\mu\}$ on the two nodes associated with pipe j, $p_{\mu(j)}$.

We now introduce the framework of *coupled learning*. Let us define two sets of constraints on the pressure values of the network. The free state p_μ^F is defined as the state where only the source nodes p_S are constrained to their values P_S, allowing p_T, p_H to obtain their steady state [i.e., to reach values that minimize the physical cost function, the dissipated power $\mathcal{P}(p_\mu; k_j)$] [Fig. 1(a)]. The clamped state p_μ^C is the state where both the source and target node pressures p_S, p_T are constrained to P_S and p_T^C, respectively, so that only the remaining (hidden) nodes p_H are allowed to change to minimize the dissipated power. The values of the dissipated power in the resulting steady states are denoted $\mathcal{P}^F(P_S, p_T^F, p_H^F; k_j)$ and $\mathcal{P}^C(P_S, p_T^C, p_H^C; k_j)$ for the free and clamped states, respectively. In the following we simplify notation by suppressing the variables in the parentheses.

A disordered flow network constrained only at its sources (free state) will generally find a steady state in which the target pressures $\{p_T^F\}$ differ significantly from the desired responses $\{P_T\}$. To concretely define the clamped state, we introduce a trainer (supervisor) that reads the free-state target node pressures and nudges them slightly away from their free-state p_T^F values by clamping them at

$$p_T^C = p_T^F + \eta[P_T - p_T^F], \qquad (3)$$

where $\eta \ll 1$ [Fig. 1(b)]. The supervisor imposes pressures on the target nodes that are a small step closer to the desired response P_T. We then propose a learning rule for the pipe conductance values [Fig. 1(c)]:

$$\dot{k}_j = \alpha\eta^{-1}\frac{\partial}{\partial k_j}\{\mathcal{P}^F - \mathcal{P}^C\}$$

$$= \frac{1}{2}\alpha\eta^{-1}\{[\Delta p_j^F]^2 - [\Delta p_j^C]^2\}, \qquad (4)$$

where α is a scalar learning rate. Note that the derivative of the physically minimized function (power dissipation $\mathcal{P}$) is taken only directly with respect to the learning degrees of freedom $\{k_j\}$ (pipe conductances), excluding derivatives of the physical variables with respect to them (e.g., $\partial p_i/\partial k_j$). In the limit of small nudges $\eta \to 0$ such derivatives cancel out when the difference between the free and clamped terms is taken [18].

We note that the learning rule of Eq. (4) is similar to the one derived for analog resistor networks by Kendall *et al.* [22]. While their equation has the same form, their definition of the clamped state is different, as they draw currents into the output nodes rather than fixing the output pressures. Their nudging is thus akin to applying a force on the output nodes. The simplest implementation of the learning rule is to iteratively apply Eq. (4) for some period of time between each nudge of the clamped boundary condition in Eq. (3). Here we focus on learning in the quasistatic limit where we completely relax the node pressure values to their steady state at each iteration to reach the minimum of the power dissipation $\mathcal{P}$.

The learning rule of Eq. (4) is manifestly local, as the conductance of a pipe k_j changes only due to the flow through that pipe. Such local learning rules may conceivably be implemented in physical pipes for which the conductance (radius) of the pipe is controlled by the current in it. Note that the network is not required to encode information about the desired state function *a priori*. That information is fed into the network by the actions of the external supervisor, which slightly nudges the target node pressures toward the desired state at every iteration. We later show how these properties of coupled learning stand in contrast to tuning algorithms based on optimization of global cost functions.

The intuition underlying coupled learning is straightforward. We wish to obtain a network with the desired target pressures $\{P_T\}$ when source pressures $\{P_S\}$ are applied. We divide the training process into small steps, each of which depends on free and clamped states. At every iteration of Eq. (3), the clamped state is initially slightly "better" than the free state, in that the node pressures at the targets $\{p_T\}$ are slightly closer to the desired pressures $\{P_T\}$ due to the applied nudge. The learning rule changes the pipe conductances $\{k_j\}$ by adding the gradient at the free state (raising the free-state power dissipation) and subtracting the gradient at the clamped state (lowering the clamped-state power dissipation). By adjusting the pipe conductances, the network response adapts to more closely reflect the clamped state. After many iterations of the learning rule in Eq. (4), $\{\dot{k}_j\}$ approach zero and the free state approaches the desired target pressures P_T. By iterating in this fashion, the network carves out a valley in the landscape of the dissipated power $\mathcal{P}$ with respect to the target node pressures $\{p_T\}$ that deepens as one approaches the desired target pressures—much as directed aging does when it is successful [11]. This intuition implies that the nudge magnitude value should be small $\eta \ll 1$, as was proposed for equilibrium propagation [18] (see the Appendix A). Such choice allows the gradual modification of the power landscape $\mathcal{P}$ so that it becomes minimal along a valley at the desired state. The approach of using small nudges has been shown to achieve excellent results on hard computational learning problems such as CIFAR-10 [23].

We test the coupled learning rule, Eq. (4), on disordered flow networks of various sizes ($N = 64$, 128, 256, 512, 1024, 2048). Each of these networks is initialized with pipes of uniform conductance $k_j = 1$ in Fig. 2(a). In each instance, we pick $M_S = 10$ nodes randomly to be the source nodes and apply the source pressures P_S, indicated by the sizes of the red dots. These source pressures are themselves chosen randomly from a Gaussian distribution $\mathcal{N}(0, 1)$. We also randomly choose $M_T = 10$ different target nodes, and desired pressures at target nodes, $\{P_T\}$, randomly from a Gaussian distribution scaled by the source pressures $\mathcal{N}(0, 0.2\sum P_S)$; their values are indicated by the sizes of the blue dots in Fig. 2(a). The free-state pressures p^F, given applied source pressures P_S, are first computed by minimizing the dissipated power. To compare the output p_T^F of the network to the desired response P_T, we use the standard error function $C = \frac{1}{2}\sum_T[p_T^F - P_T]^2$. Untrained networks have a large error, and training should gradually decrease it. The network achieves the function perfectly when the error vanishes.

We now consider the clamped state by nudging the target nodes toward their desired values, setting their pressures to $\{p_T^C = p_T^F + \eta[P_T - p_T^F]\}$, with $\eta = 10^{-3}$. We minimize power dissipation with respect to the hidden nodes p_H and then update the conductance values (learning degrees of
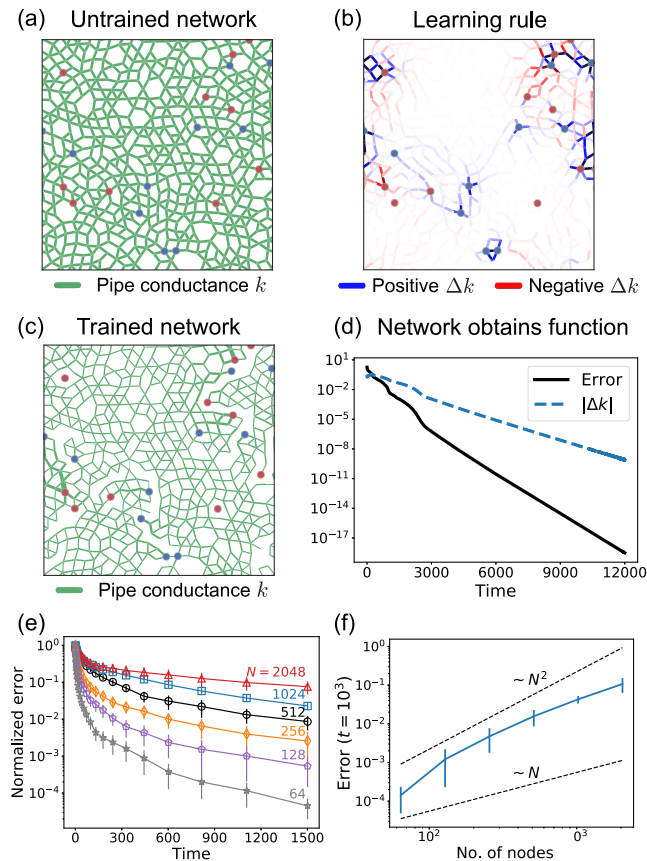
FIG. 2.   Training flow networks with coupled learning. (a) An untrained disordered flow network with uniform conductance at all pipes $k_i = 1$, as indicated by uniform thicknesses of the green edges. The ten red and blue nodes correspond to the source and target nodes with dot sizes indicating the magnitudes of the source pressures $\{P_S\}$ and *desired* target pressures $\{P_T\}$. (b) In each step, conductance values are modified using Eq. (4), according to the difference in flow between the free and clamped states. This process is applied iteratively. (c) After training, the network conductance values, indicated by the thicknesses of the green edges, are considerably changed compared to the initial network shown in (a). (d) During training of a network ($N = 512$ nodes), the pressure values of the target nodes approach the desired values, as indicated by the exponentially shrinking error (black solid line). The desired target values $\{P_T\}$ are reached when the error is small; the modification of the conductance in each time step, Δk (blue dashed line), vanishes exponentially as well. (e) We train multiple networks of different sizes $N = 64$–2048, and find that all can be trained successfully with coupled learning. Error bars indicate the variation with initial network and choice of sources and targets. In all cases, errors decay exponentially, yet larger networks converge slower. (f) Picking a certain time $t = 10^3$, we find that the error scales up with system size as a soft power between 1 and 2.

freedom) according to Eq. (4) with a learning rate of $\alpha = 5 \times 10^{-4}$. Figure 2(b) shows the change of conductance of each edge at the first iteration of learning, with blue (red) signifying positive (negative) conductance changes.

This process constitutes one step of the training process; at the end of each step, we compute the error function C [Fig. 2(d)]. The difference between the obtained targets and the desired ones decreases exponentially by many orders of magnitude during the training process, reaching machine precision. This result demonstrates the success of the coupled learning approach. We see that the magnitude of the change in the conductance vector, $|\Delta k| = \sqrt{\sum_j \Delta k_j^2}$, calculated for each step of the training process, also decreases exponentially during training [blue dashed line in Fig. 2(d)]. This result shows that the learning process is adaptive—it slows down as it approaches good solutions. The final trained network is displayed in Fig. 2(c), with edge thicknesses indicating conductance. The pipes of the trained network have changed considerably compared to the initial one shown in Fig. 2(a), with some pipes effectively removed (with conductances near zero).

The results of applying the training protocol to networks of different sizes, for different initial networks and choices of the source and target nodes and their pressure values, are shown in Figs. 2(e) and 2(f), where errors are rescaled by the initial error for each network. Our learning algorithm is generally able to train the networks to exhibit the desired responses, successfully decreasing the initial error by orders of magnitude. We find that networks of different size converge on good solutions at different rates, with the error at a particular chosen time $t = 10^3$ scaling roughly as a power law $C(N, t) \sim N^q$ (with power q in the range 1–2). We note that networks of different sizes may not be equivalent, as training may depend on idiosyncratic details, such as particular distances between sources and targets, or other geometrical features. We leave detailed exploration of the effects of network size and source-target geometry to future study.

It is noteworthy that flow networks are linear, so that the mapping between the sources to targets is always linear $p_T = A(k)P_S$ (A is a $M_S \times M_T$ matrix that depends on the conductance values). Networks which contain hundreds of edges have many more conductance values than components of A so that there are far more degrees of freedom than constraints. While this argument suggests our flow networks are overparametrized and should always succeed in learning, we stress that not all linear transformations are possible; pressure values everywhere in the network are weighted averages of their neighbors (due to Kirchhoff's law). More importantly, the linear transformation is limited because all conductance values must be non-negative (see Appendix D). As a result, flow networks cannot implement any desired linear mapping between the inputs and outputs, and nonzero errors are expected for certain tasks after training. It was previously shown that the likelihood of flow networks to successfully learn a set of tasks depends on network size [6], even when trained with gradient descent. Therefore, we expect that training larger networks for a given task is more likely to succeed

due to overparametrization, at the expense of slower convergence rates.

Furthermore, while computational neural networks are often initialized with random (e.g., Gaussian distributed) weights to compensate for their symmetries [24,25], we find that our disordered networks can be trained successfully with uniform $k_j = 1$ initialization. Indeed, our tests with initial weights of the form $k_j = \mathcal{N}(1, \sigma^2)$, with $0 < \sigma \le 0.5$, have yielded qualitatively similar results to those shown in Fig. 2(b), with respect to both training time and accuracy.

Recently, a similar set of ideas [22,23] based on equilibrium propagation was independently proposed to train electric resistor networks. This approach is very similar to our framework. The difference is that Refs. [22,23] used a nudged state defined by injecting currents to target nodes, instead of by clamping voltages as in our approach. Their method converges to gradient descent in the limit $\eta \to 0$. While our method does not, we find that coupled learning and gradient descent give very similar results in our trained networks (see Sec. IV). Our focus in this paper is also somewhat different from theirs; we showcase coupled learning in general physical networks in which the system naturally tends to minimize a physical cost function with respect to the physical degrees of freedom, and emphasize considerations regarding the implementation of coupled learning in experimental physical networks.

B. Coupled learning in general physical networks

We now formulate coupled learning more generally so that it can be applied to general physical networks. Consider a physical network with nodes indexed by μ and edges indexed by j. As in the special case described earlier, $\{x_\mu\}$ are physical degrees of freedom (e.g., node pressure for a flow network). Network interactions depend on the learning degrees of freedom on the network edges $\{w_j\}$ (e.g., pipe conductance in flow networks). We restrict ourselves to the athermal case where, given some initial condition $x_\mu(t = 0)$, the physical degrees of freedom evolve to minimize a physical cost function E (e.g., the power dissipation for a flow network) to reach a local or global minimum $E(x_\mu; w_j)$ at x_μ, where both E and $\{x_\mu\}$ depend on $\{w_j\}$.

We define the free state x_μ^F where the source variables x_μ alone are constrained to their values, and x_T, x_H equilibrate [i.e., minimize $E(x_\mu; w_j)$]. At the clamped state x_μ^C both source and target variables x_S, x_T are constrained, so only the hidden variables are equilibrated. An external trainer (supervisor) nudges the target nodes by a small amount η toward the desired values X_T,

$$x_T^C = x_T^F + \eta[X_T - x_T^F], \qquad (5)$$

with $\eta \ll 1$. The general form of coupled learning is then

$$\dot{w} = \alpha\eta^{-1}\partial_w\{E^F(x_S, x_T^F) - E^C(x_S, x_T^C)\}, \qquad (6)$$

where α is again the learning rate. We thus find a general learning rule similar to equilibrium propagation [18], with the understanding that only the direct derivatives ($\partial_w E$) are performed, excluding the physical state derivatives $\partial_w x^F, \partial_w x^C$ which cancel out. Given any physical network with a known physical cost function $E(x_\mu; w_j)$, one can derive the relevant coupled learning rule directly from Eq. (6). Coupled learning is performed iteratively until the desired function is achieved. Note that in physical networks, E can generically be partitioned as a sum over edges $E = \sum_j E_j(x_\mu(j); w_j)$. Each term E_j for edge j depends only on the physical degrees of freedom attached to the two nodes $\mu(j)$ that connect to edge j. The learning rule of Eq. (6) is thus guaranteed to be local, in contrast to many design schemes relying on the optimization of a global cost function. Moreover, the network is not required to encode any information about the desired response. This information is fed to the network by the trainer, which nudges the target physical degrees of freedom closer to the desired state at every iteration of the learning process. Note that while our discussion so far and Fig. 1 are restricted to a physical network with binary edges (i.e., binary physical interactions), coupled learning does not assume binary interactions and is valid for arbitrary n-body potentials. For more information on the general coupled learning rule, see Appendix B.

C. Elastic networks

To demonstrate the generality of our coupled learning framework, we apply it to another physical system, central-force spring networks. Here we have a set of N nodes embedded in d-dimensional space, located at positions $\{x_\mu\}$. The nodes are connected to their neighbors in the network by linear springs, each having a spring constant k_j and equilibrium length ℓ_j. The energy of a spring, labeled j, depends on the strain of that spring, $E_j = \frac{1}{2}k_j(r_j - \ell_j)^2$, where r_j is the Euclidean distance between the two nodes connected by the spring. The physical cost function in this case is the total elastic energy of the network, given by $E = \sum_{j\in\text{springs}} E_j$. We choose source and target springs, whose lengths are r_S and r_T, and train the network so that an application of source edge strains $\{R_S\}$ gives rise to desired target edge strains $\{R_T\}$.

In contrast to flow networks, spring networks are non-linear in the physical variables $\{x_\mu\}$, due to the nonlinearity in the Euclidean distance function. Moreover, while the spring constants $\{k_j\}$ are formally equivalent to conductances in flow networks, the equilibrium lengths $\{\ell_j\}$ have no direct analog in flow networks. These extra variables are
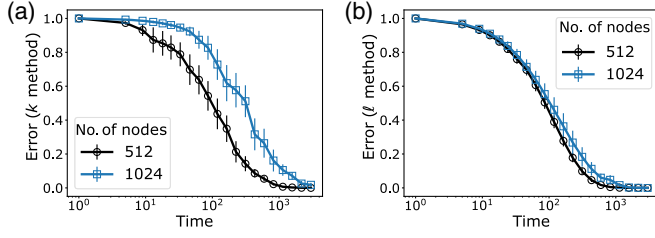
FIG. 3. Training spring networks by modifying (a) spring constants or (b) rest lengths. The error in the target edge lengths is shown as a function of the number of training iterations for $N = 512$ (black circles) and $N = 1024$ (blue squares) networks. Training by coupled learning is successful for either the stiffness or rest length degrees of freedom.

additional learning degrees of freedom that we can adjust in addition to the spring constants.

The free and clamped states of the spring network are applied similarly to the previous example, with the exception that we define the source and target boundary conditions on the edges of the network rather than the nodes, demonstrating that the coupled learning framework can be applied in either case. Next, we apply the coupled learning rule (6) to obtain two separate learning rules, one for the spring constants k_j, the other for the rest lengths ℓ_j:

$$\dot{k}_j = \frac{\alpha}{\eta} \frac{\partial}{\partial k_j} \{E^F - E^C\}$$
$$= \frac{\alpha}{2\eta} \{[r_j^F - \ell_j]^2 - [r_j^C - \ell_j]^2\},$$
$$\dot{\ell}_j = \frac{\alpha}{\eta} \frac{\partial}{\partial \ell_j} \{E^F - E^C\}$$
$$= \frac{\alpha}{\eta} k_j \{[r_j^F - \ell_j] - [r_j^C - \ell_j]\}. \tag{7}$$

Learning in elastic networks can be accomplished through modification of the spring constants, rest lengths, or a combination of both. As before, Eq. (7) gives local learning rules, where each spring only changes due to the strain on that particular spring. To test these learning rules, we train elastic networks with $N = 512$ and $N = 1024$ nodes with multiple choices of 10 random source strains and 3 random target strains. Elastic network calculations were performed using a specialized benchmarked set of computational tools, used earlier for research on tuned networks [5,26]. The success of the training is again assessed using the error $C = \frac{1}{2} \sum_T [r_T^F - R_T]^2$, measuring deviation from the desired target lengths. We find that regardless of whether the network learns by modifying its spring constants [Fig. 3(a)] or rest lengths [Fig. 3(b)], the networks are consistently successful in achieving the desired target strain, reducing the error C by orders of magnitude. Larger networks take slightly longer to learn, but achieve similar success. We find that the rest-length-based learning

rule gives somewhat more consistent learning results for different networks and source-target choices, as evidenced by smaller error bars in Fig. 3(b). Recently, a rule similar to that of Eq. (7) was used to prune edges in elastic networks to obtain desired allosteric responses [27]. It is notable that Ref. [27] uses a "symmetrized" version of the learning rule, where the free state is replaced with a negative clamped state, where the output nodes are held farther away from their respective desired value.

D. Supervised classification

In Secs. II A and II C we trained networks to exhibit functions inspired by allostery in biology, obtaining a desired map between one set of sources (inputs) and one set of targets (outputs). Here we train flow networks to display a function inspired by computer science, namely, the ability to classify images. In classification, a network is trained to map between multiple sets of inputs and outputs. In order to use our coupled learning protocol for simultaneous encoding of multiple input-output maps, we slightly modify the training process in the spirit of stochastic gradient descent [28]. Each training image provides an input, and the correct answer for that image is the desired output. Training examples are sampled uniformly at random, and the coupled learning rule of Eq. (4) is applied accordingly.

A simple example of a classification problem, often used to benchmark supervised machine learning algorithms, is to distinguish between labeled images of handwritten digits (MNIST [29]). Each image corresponds to a particular input vector (e.g., pixel intensity values), while the desired output is an indicator of the digit displayed. Typically an algorithm is trained on a set of example images (training set), while the goal is to optimize the classification performance on unseen images (test set).

Here, we train flow networks to distinguish between images of two digits (0 and 1). We pick 50 images of each digit (class) from the MNIST database to serve as a training set, and an additional 50 images of each digit as a test set. Instead of using the images themselves, we first carry out a principal component analysis of all MNIST images, and train the network using the top 25 principal components of the training set images. The inputs (source pressures) are given by these principal components of the training images for a randomly selected set of 25 source nodes. We additionally pick 2 random output (target) nodes, one for each digit. The desired output for a training example corresponding to a 0 is for the first output node, the 0 target node, to have high pressure ($p_{\cdot 0'} = 1$), and for the second output node, the 1 node, to have no pressure ($p_{\cdot 1'} = 0$). The target pressures are reversed when an example corresponding to the digit 1 is chosen so that the 1 node has $p_{\cdot 1'} = 1$ and the 0 node has $p_{\cdot 0'} = 0$. At each iteration of the learning rule, the network is presented with a single training image, sampled uniformly at random from
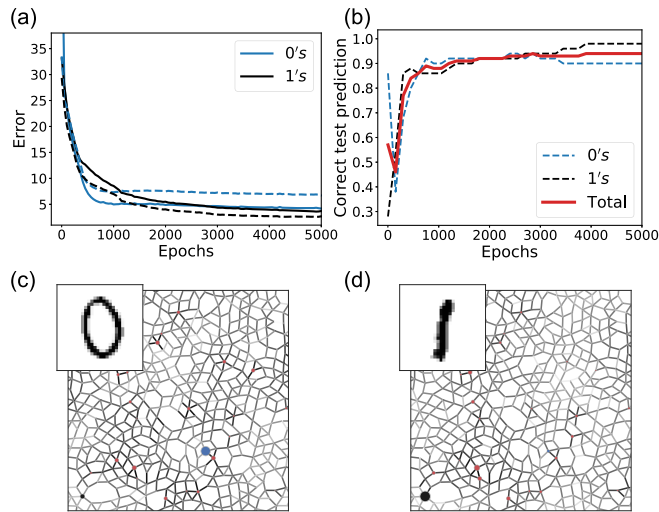
FIG. 4. Classification of MNIST digits (0's and 1's). (a) The classification error (value of cost function) [Eq. (11)] for the digits 0 (blue) and 1 (black) versus the number epochs, where each epoch consists of 100 iterations of the learning rule with a different randomly selected training image presented at each iteration. Training error is indicated by solid lines and test error by dashed lines. (b) Prediction accuracy on test images (fraction of correct predictions made by the network for digits 0 and 1) versus number of epochs. Total test accuracy shown in red. (c),(d) Response of the network when presented with the image at the top left. The power dissipated at each edge is indicated by the intensity of the edge color. while the pressures of target nodes are labeled with blue (0) and black (1) circles. When a 0 image is shown, the 0 target node has high pressure (blue circle) and the 1 target node has low pressure; when a 1 image is shown, the 1 target node has high pressure (black circle) and the 0 target node has low pressure.

the training set, and is nudged toward the desired output for that image. A training epoch is defined as the time required for the network to be presented with 100 training examples.

The error between the desired and observed behavior is shown in Fig. 4(a) for each of the digits. The solid lines represent errors for the training set, while the dashed lines show errors for the test set. We see that the network not only reduces the training error, but also successfully generalizes, reducing the test errors as well. While the pressure at the output nodes is continuous, we want the network to make a discrete choice about the label of the shown image. We say that the predicted label is given by the target node with the larger pressure. With this definition, we see that the classification accuracy of the network starts at about 50% and improves dramatically during training [Fig. 4(b)]. After training, the overall training accuracy reaches 98%, while the test accuracy is 95%. This learning performance is almost as good as that of simple machine learning algorithms such as logistic regression, which for the same data yields a training accuracy of 100% and a test accuracy of 98%. Note that

we did not tune hyperparameters to achieve the listed accuracies; such tuning might well improve the performance of our algorithm. In Figs. 4(c) and 4(d), the network response to two select input images is shown. When the source nodes are constrained with pressure values corresponding to an image of a 0, the 0 target node has high pressure (blue) while the 1 target node has very low pressure. The opposite response occurs when an image of a 1 is applied as input.

As discussed above, we used a training approach in the spirit of stochastic gradient descent, training the network with one training example at a time. We note that it may be experimentally possible to train the network using batch gradients, presenting many training examples in quick succession, so that the learning rule averages the conductance update over these examples. A similar effect can be achieved by maintaining the rate of example presentation, yet slowing down the learning rate α.

III. EXPERIMENTAL CONSIDERATIONS

In the previous section we introduced the coupled learning rules that respect the physical requirements of locality and the fact that the desired response cannot be encoded in the network *a priori*. We have demonstrated computationally that flow and mechanical networks successfully learn desired behaviors using such learning rules. For networks to learn autonomously in a physical setting, the network must be able to implement the coupled learning rules as the physical dynamics of their learning degrees of freedom. Furthermore, an external supervisor needs to be able to clamp the target nodes as needed. For fluid flow networks, coupled learning requires the coupling of pipe conductance values to the flow inside the pipe. For example, constriction of pipes can be achieved by entrainment of sediment in the flow, and its deposition inside the pipes [30,31]. These dynamics could support a one-sided learning rule as described in Sec. III A. In elastic networks, either the stiffnesses or rest lengths of bonds need to change for learning to be possible. Several classes of materials are known to adapt their stiffness in response to applied strain, e.g., ethylene vinyl acetate [32] and thermoplastic polyurethane [33]. The former has been used to train materials for auxetic behavior using directed aging [9].

In the following we describe some potential experimental difficulties in implementing coupled learning in physical networks and how they may be overcome. We first discuss an approximation to the coupled learning rules, allowing for a simpler implementation in flow networks (Sec. III A). Then, we describe the effects of plausible experimental issues, such as noisy physical degrees of freedom or drifting learning degrees of freedom (Sec. III B), and the effect of network size on the physical relaxation assumed in coupled learning (Sec. III C).

A. Approximate learning rules

Consider the flow networks described earlier. To implement the learning rule of Eq. (4), one requires pipes whose conductance can vary in response to the dissipated flow power through them. There are two major issues in implementing this learning rule in a physical flow network. First, the learning rule of Eq. (4) for the conductance on edge j depends not simply on the power through edge j but on the difference between the power when the free-boundary condition and the power when the clamped-boundary condition are applied. This is difficult to handle experimentally because it is impossible to apply both sets of boundary conditions simultaneously to extract the difference. One could try to get around this by alternating between the free- and clamped-boundary conditions during the process, but then one runs into the difficulty that the sign of conductance change is opposite for the two types of boundary conditions. In other words, the same power through edge j in the free and clamped states would need to induce the opposite change in conductance. Alternating the sign of the change in conductance along with the boundary conditions poses considerable experimental difficulty. The second hurdle is that Eq. (4) requires that pipes must be able to either increase or decrease their conductance. Pipes whose conductance can change in only one direction (e.g., decreasing k by constriction) are presumably easier to implement experimentally.

To circumvent these difficulties, we seek an appropriate approximation of Eq. (4) (see Appendix C for more details) that is still effective as a learning rule. We first define a new hybrid state of the system, named the δ state, in which the power dissipation is minimized with respect to the hidden node pressures p_H with the constraints $p_S = 0$, $p_T = -\eta(P_T - p_T^F)$.

Note that this state corresponds to constraining the source and target nodes to the desired pressure differences between the free and clamped states according to the original approach. Now, we may expand the clamped-state power in series around the free state, to obtain a new expression for the learning rule in terms of the δ state and free state:

$$\dot{k}_j \approx 2\alpha\eta^{-1}\Delta p_j^F \Delta p_j^\delta.$$

Ideally only the δ state would be involved in the learning rule and not the free state. Let us simplify this rule by only accounting for the sign of the free pressure drop, $\mathrm{sgn}(\Delta p_j^F)$, returning ± 1 depending on the sign of flow in the free state, $\dot{k}_j \approx 2\alpha\eta^{-1}\mathrm{sgn}(\Delta p_j^F)\Delta p_j^\delta$.

The resulting learning rule, while only depending on the pressures at the δ state, can still induce either positive or negative changes in k. To avoid this, we may impose a step function, that only allows changes in one direction (e.g., only decreasing k):
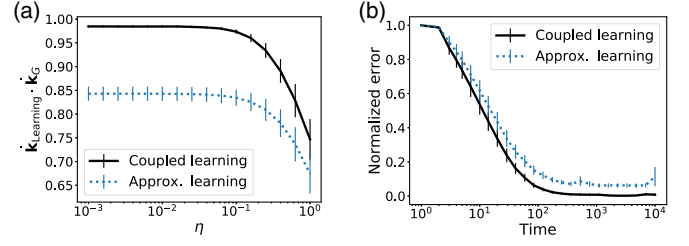
FIG. 5. Training a flow network (512 nodes, 10 random sources, and 3 random targets) with an approximate, experimentally viable learning rule. (a) The full coupled learning rule [Eq. (4)] and the approximate learning rule [Eq. (8)] are compared to the gradient of the cost function for different values of η [see Fig. 7(c)]. While the full learning rule is superior, there is still a significant correlation between optimal tuning and the approximate rule. (b) The network may be trained using both the full and approximate rule, at similar timescales. The approximate learning rule, being much more restricted, often saturates at a higher error level compared to the full rule (averaged over 17 random source-target sets).

$$\dot{k}_j = -2\alpha\eta^{-1}\Theta(-\Delta p_j^F \Delta p_j^\delta)|\Delta p_j^\delta|. \tag{8}$$

The intuition behind this learning rule is relatively simple: if for a particular pipe $\mathrm{sgn}(\Delta p_j^F)\Delta p_j^\delta < 0$, then too much flow is going through the pipe in the free state. Thus, one could improve the functional result by decreasing the conductance of that pipe.

This learning rule may be simpler to implement physically, but is it a sufficiently close approximation of the original rule? In Fig. 5 we train a network (512 nodes, with 10 random sources and 3 random targets) using the learning rules of Eqs. (4) and (8). In Fig. 5(a), the learning rules are compared to the tuning algorithm [Eq. (13)] by taking the dot product of these modification [similar to Fig. 7(c)]. While the approximate learning rule has a lower dot product with the optimal local tuning for all values of η compared to the full learning rule, the correlations are still very significant. We thus expect the approximate learning rule to train the network successfully. Indeed, we find that when the approximate learning rule is applied iteratively, the learning process is successful, yielding a decrease of the error C of at least an order of magnitude [Fig. 5(b)]. While the approximate training typically plateaus at a higher error level compared to the full coupled learning rule, the approximate learning rule can successfully train flow networks to exhibit random source-target responses.

B. Noise in physical or learning degrees of freedom

So far, we have assumed that coupled learning modifies the learning degrees of freedom given precise knowledge of the free and clamped states [e.g., $\mathcal{P}^F$, $\mathcal{P}^C$ in Eq. (4)]. However, in any physical implementation of a network there will be errors in the measurements of these states. Suppose that measurement of the dissipated power at every
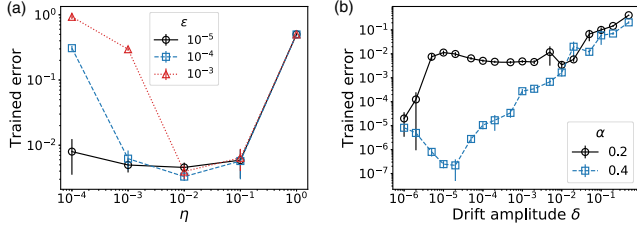
FIG. 6. Learning with noisy measurements and diffusing learning degrees of freedom. (a) When noise is present in the free- or clamped-state measurements, small nudges impede learning. Training results can be improved with larger $\eta < 1$ nudges. (b) Drifting in the learning degrees of freedom impedes learning, but its effects can be mitigated by increasing the learning rate α.

edge is subject to additive white noise of magnitude ϵ, noted $e_j \sim \mathcal{N}(0, \epsilon^2)$. The learning rule would then contain an extra term due to the noise:

$$\dot{k}_j \approx \frac{1}{2}\alpha\eta^{-1}\{[\Delta p_j^F]^2 - [\Delta p_j^C]^2 + e_j\}. \qquad (9)$$

It is clear from Eq. (9) that if the error magnitude ϵ is larger than the typical difference between the free and clamped terms, the update to the conductance values would be random and learning will fail. As we show in Appendix B, the difference between the free and clamped term scales with the nudge amplitude $[\Delta p^F]^2 - [\Delta p^C]^2 \sim \eta^2$. Put differently, when the nudge is small ($\eta \to 0$), the free and clamped states are nearly identical, so that their difference is dominated by noise. This raises the possibility that increasing the nudge amplitude η used, i.e., nudging the clamped state closer to the desired state, will increase the relative importance of the learning signal compared to the noise, allowing the network to learn despite the noise.

To test this idea, we train a flow network of $N = 512$ nodes on a random task with 10 source and 10 target nodes [the same training problem shown in Figs. 2(a)–2(d)]. While training, we inject white noise of given magnitude ϵ according to Eq. (9). As suggested, we find that these "measurement" errors primarily spoil learning at low nudge values, so that increasing η might have a beneficial effect on learning [Fig. 6(a)]. We conclude that experimental settings with finite measurement errors induce a trade-off in the nudge parameter η. Nudges should be small so that the learning rule gives a good estimate of the desired gradients, yet they cannot be too small, or the learning will be dominated by noise.

Another source of error in an experimental realization of learning networks may be unwanted fluctuations or drift in the learning parameters. For example, it is known that natural neural networks undergo synaptic fluctuations, even in adults and in the absence of synaptic activity [34–36]. In synthetic contexts, engineering materials with controllable parameters in a wide range, e.g., variable stiffness in shape memory alloys [37], can lead to an increased volatility in these controlled parameters [38].

Drifting in the learning degrees of freedom during the training process is expected to impede learning. To study its effects, we assume that the learning degrees of freedom fluctuate with some magnitude δ for all edges. This gives rise to additive white noise $d_j \sim \mathcal{N}(0, \delta^2)$ in the learning rule:

$$\dot{k}_j \approx \frac{1}{2}\alpha\eta^{-1}\{[\Delta p_j^F]^2 - [\Delta p_j^C]^2\} + d_j. \qquad (10)$$

In contrast to the noise in the physical degrees of freedom discussed earlier, such diffusion cannot be controlled by increasing the nudge amplitude. However, one can increase the learning rate α to hasten the learning dynamics compared to diffusion (keeping in mind that to maintain convergence, learning cannot be made too fast [4]).

Once more, we train a flow network ($N = 512$ nodes) on a task with 10 random sources and 10 random targets [the same task as in Figs. 2(a)–2(d)]. The network is trained with different diffusion magnitudes δ and two different learning rates $\alpha = 0.2, 0.4$. We find that given constant diffusion rates, training is more successful at a higher learning rate, as expected [Fig. 6(b)]. These results show that unavoidable experimental issues such as noise in the physical and learning degrees of freedom can be mitigated by controlling learning hyperparameters such as the nudge amplitude η and the learning rate α.

C. Physical relaxation

In this work, we have taken the quasistatic limit of coupled learning, so that before computing either the free or the clamped state, the network first reaches its stable equilibrium (or steady state). In other words, we explored the limit where learning is much slower than the physical dynamics [39]. Physical networks, however, take finite time to relax to their equilibrium, which does not scale with the amplitude of perturbation η (in the linear response regime). Therefore, using small nudges in general does not help with accelerating the physical relaxation of the network toward its equilibrium states.

Moreover, the nudging between the free and clamped states induces a local perturbation, whose information content must propagate to the entire network before it can equilibrate. In both flow networks and elastic spring networks, the time for this information propagation is set by the speed of sound. In flow networks, the speed of sound depends on the compressibility of the fluid. Similarly, elastic spring networks have a finite speed of sound depending on their node masses, edge spring constants, and network structure. This information propagation time implies a linear lower bound on the scaling of the relaxation

time with system size. Depending on the physics of the network and its architecture, relaxation time might scale more slowly with size. For example, in large branched dissipative flow networks the response time to a perturbation scales quadratically [40].

Consider a flow or elastic network of linear dimension L, trained by waiting time τ for relaxation before updating the learning degrees of freedom in Eq. (6). Because of the aforementioned considerations, training a larger network of linear dimension L' would require waiting $\tau' \approx \tau f(L'/L)$ (with f a faster than linear function) for similar relaxation, so that the overall training time should scale with the physical length of the network. Note that this physical time scaling is different from the scaling of learning convergence with system size discussed earlier. At the limit of small enough learning rates α, when the learning dynamics effectively approximate gradient descent on the cost function, we argue that rescaling the learning rate in Eq. (6) by $\alpha' = \alpha f(L'/L)$ would counteract this increase in training time. However, the learning rate cannot increase without bound for two reasons. First, at high learning rates, learning may not converge as the gradient step may overshoot the minimum of the cost function, as is often the case in optimization problems [4]. Furthermore, increasing the learning rate compared to the physical relaxation rate would eventually break the quasistatic assumption in Eq. (6). We leave the effects of breaking the assumption of quasistaticity to future study.

IV. COMPARISON OF COUPLED LEARNING TO OTHER LEARNING APPROACHES

To emphasize the advantages of our local supervised learning approach, we compare it to global supervised learning, where we adjust the learning degrees of freedom to minimize a learning cost function (or error function). Such a rational tuning strategy was previously demonstrated to be highly successful [6] and is fundamental to supervised machine learning [4]. Here we compare coupled learning and tuning in the contexts of flow and elastic networks.

One usually defines the learning cost function as the distance between the desired response P_T and the actual target response p_T^F. A commonly chosen function is the L_2 norm (Euclidean distance):

$$\mathcal{C} \equiv \frac{1}{2}[p_T^F - P_T]^2. \tag{11}$$

A straightforward way to minimize this function with respect to the learning degrees of freedom (pipe conductance values $\{k_j\}$) is to perform gradient descent. In each step of the process, the source pressures $\{P_S\}$ are applied, the physical cost function (total dissipated power) is minimized to obtain the pressures $\{p_T^F\}$, the gradient of the learning cost function is computed, and the conductances $\{k_j\}$ are changed according to

$$\partial_k\mathcal{C} = [p_T^F - P_T] \cdot \partial_k p_T^F,$$
$$\dot{\boldsymbol{k}}_G = -\alpha\partial_k\mathcal{C}, \tag{12}$$

where $\dot{\boldsymbol{k}}_G$ denotes the change in the pipe conductance predicted by the tuning (global supervised learning) strategy. Note that such a process cannot drive autonomous adaptation in a physical network—Eq. (12) cannot be a physical learning rule—for two fundamental reasons. First, the update rule is not local. Generally, the target pressure p_T^F depends on the conductance of every pipe in the network, so each component of $\partial_k\mathcal{C}$ contains contributions from the entire network. The modification of each pipe conductance depends explicitly on the currents through all of the other pipes, no matter how distant they are. Second, the tuning cost function $\mathcal{C}$ depends explicitly on the desired response P_T. Thus, if the network computes the gradient, it must encode information about the desired response. A random disordered network is not expected to encode such information *a priori*. Together, these two properties of the tuning process imply that this approach requires both computation and the modification of pipe conductance values by an external designer—it cannot qualify as a physical, autonomous learning process.

The second point above, that the tuning cost function depends explicitly on the desired behavior P_T, deserves further discussion. In coupled learning, the desired target values P_T do not appear explicitly in the learning rule of Eq. (4), but do appear explicitly in the definition of the clamped state [Eq. (3)]. This is a subtle but crucial distinction between coupled learning and tuning. In coupled learning, a trainer (supervisor) imposes boundary conditions only on the target nodes. The physics of the network then propagates the effect of these boundary conditions to every edge in the network. Then, each edge independently adjusts its learning degrees of freedom taking into account only its own response to the applied boundary conditions. In other words, the boundary conditions, imposed by the trainer, depend on the desired response $\{P_T\}$, but the equations of motion of the pipe conductance values $\{k_j\}$ themselves do not require knowledge of $\{P_T\}$ once the boundary conditions are applied. The trainer needs to know the desired network behavior, but the network itself does not. In the tuning process, by contrast, the pipe conductance values evolve according to an equation of motion that depends explicitly on $\{P_T\}$. Thus, tuning a physical network requires external computation of $\partial_k\mathcal{C}$ and the subsequent modification of the learning degrees of freedom by an external agent.

The difference between local and global supervised learning has fueled long-standing debates on how biological networks learn, and their relation to computational tuning approaches such as machine learning algorithms [41]. While natural neural systems are complicated and can perform certain computations, simple physical networks

such as the ones studied here definitely cannot. In order for these networks to adapt autonomously to external inputs and learn desired functions from them without performing computations, we need a physically plausible learning paradigm such as the coupled learning framework presented here.

We now directly compare results of the two learning methods in flow and elastic networks. First, we use both methods to train flow networks of size $N = 512$ nodes on the tasks described in Sec. II A [Fig. 7(a)]. We find that, on average, coupled learning is as efficient as tuning (gradient descent) for training these networks. Indeed, for the problem shown in Fig. 2(d), coupled learning actually converges faster than tuning.

To explore the comparison further, we denote the conductance modification prescribed by coupled learning as $\dot{\boldsymbol{k}}_{\mathrm{CL}}$ [Eq. (4)] and the conductance modification dictated by global supervised learning, or tuning, as $\dot{\boldsymbol{k}}_G$ [Eqs. (11) and (12)]. We argue theoretically in Appendix B that coupled learning leads to modifications to the learning degrees of freedom k_j that are similar to those obtained by tuning using gradient descent [Eq. (12)]. Therefore, we compare the two modification vectors directly by taking a dot product between them (after normalization):

$$\dot{\boldsymbol{k}}_{\mathrm{CL}} \sim \partial_{\boldsymbol{k}} \{ \mathcal{P}^F - \mathcal{P}^C \},$$
$$\dot{\boldsymbol{k}}_G \sim -\partial_{\boldsymbol{k}} \mathcal{C}. \qquad (13)$$

These two modification vectors are compared in Figs. 7(b)–7(d) for flow networks with $N = 512$ nodes, trained with 10 random sources and varying numbers of random targets. Most notably, we observe that the two vectors are indeed nearly identical (the dot product of the normalized vectors is close to unity) for sufficiently small nudges η [Fig. 7(c)]. This is remarkable as the vectors reside in the very high (~ 1000) dimensional space of conductance values $\boldsymbol{k}$ of the network. Thus Fig. 7 shows that the local process of coupled learning follows a trajectory in conductance space that is very similar to the trajectory followed during rational tuning, which is a global supervised learning process [although we do note fluctuations in the dot product; see the enlarged scale in Fig. 7(b) and Appendix D]. For small nudges $\eta \lesssim 0.1$, we observe a plateau in this dot product, so the precise choice of η is not important [choosing small $\eta \to 0$ leads to a well-defined limit of the coupled learning rule of Eq. (13); see Appendix B]. On the other hand, choosing large $\eta \sim 1$ (as in Hebbian contrastive learning) leads to significant differences between the two processes. We also note that for more complicated functions (more targets to satisfy), the dot product slightly decreases. This effect is exhibited in Fig. 7(d) for networks of two sizes. However, even with extra functional complexity, coupled learning yields results that are remarkably similar to tuning.
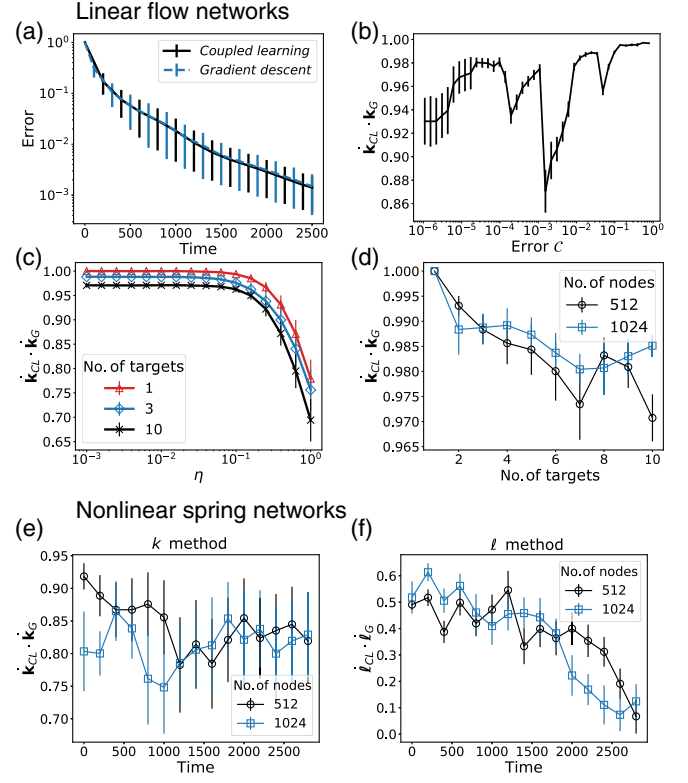

FIG. 7. Coupled learning (local supervised learning) compared to tuning (global supervised learning) in linear flow and nonlinear elastic networks. (a) Direct comparison between training traces in flow networks shows that coupled learning is as successful as gradient descent. (b) The dot product of the modification vector $\Delta \boldsymbol{k}$ for coupled learning and tuning (gradient descent) during training undergoes fluctuations, yet remains high. (c) At the beginning of training, we compute the dot product of the modification vector $\Delta \boldsymbol{k}$ for coupled learning and tuning as a function of the nudge variable η for systems with 1 (red triangles), 3 (blue diamonds), and 10 (black crosses) target nodes. The two methods predict very similar results for $\eta \ll 1$. (d) For $\eta = 10^{-3}$, coupled learning yields similar results to tuning for networks with $N = 512$ (black circles) and $N = 1024$ (blue squares), as a function of the number of target nodes. (e),(f) Dot product between the normalized predictions of coupled learning and gradient descent in nonlinear elastic networks. We find that coupled learning and gradient descent are largely aligned during the training of elastic networks using the stiffness and rest lengths degrees of freedom ($N = 512, 1024, M_S = 10, M_T = 3$).

Unlike flow networks, elastic spring networks are fundamentally nonlinear, so it is interesting to explore whether coupled learning produces similar modifications to the learning degrees of freedom as tuning (gradient descent). In Figs. 7(e) and 7(f), we show the alignment between the coupled learning and tuning. We find that the alignment between these modification vectors is still very significant, especially for training with the stiffness degrees of freedom k_j. We further find that the alignment is reduced during training as the network approaches low-error solutions, in particular for training the rest lengths ℓ_j. While

this result is interesting and suggests that coupled learning may differ significantly from tuning in certain nonlinear networks, we note that training the network is successful regardless of the deviation between the methods. We leave more detailed exploration of the alignment between possible local learning rules to future study.

A physical learning process called "directed aging" has recently been introduced to create mechanical metamaterials from mechanical networks [9–11] and self-folding sheets [13]. The strategy exploits the idea that stressed bonds in elastic networks tend to weaken (age) over time. During training, the network is nudged in a similar fashion to the coupled learning process. References [9,10] consider two different classes of learning degrees of freedom in central-force spring networks. In one case, the "k model," the stiffnesses k_j of the springs are modified, while in the "ℓ model" rest lengths $\ell_{0,j}$ are modified. The clamped state can correspond either to Eq. (5) with $\eta = 1$ or to a periodically varying amplitude $\eta = \sin \Omega t$. In the latter case, the oscillation is performed quasistatically in the sense that the physical degrees of freedom are relaxed completely at each time step.

In elastic networks, the directed aging k-model and ℓ-model learning rules are

$$\dot{k}_j = -\frac{\alpha}{2}[r_j^C - \ell_j]^2,$$
$$\dot{\ell}_j = -\alpha k_j [r_j^C - \ell_j], \tag{14}$$

where the learning degrees of freedom (stiffness or rest length) evolve in response to the clamped boundary conditions. Such dynamics have the effect of lowering the elastic energy of the desired state, and thus the response of the network to the specified inputs is expected to improve. Indeed, directed aging was shown to be successful in training elastic networks with nearly marginal coordination, so that they lie just above the minimum number of edges per node ($Z_c = 2d$) required for mechanical stability in d dimensions. Note that because the clamped physical cost function E^C can be written as a sum over individual costs of edges j, the directed aging rule is local as it is for coupled learning. Directed aging therefore corresponds to a physical learning rule, as has been demonstrated experimentally [9].

However, directed aging fails to achieve the desired target response in many instances. In particular, it is observed that either flow networks or highly coordinated elastic networks cannot be trained by directed aging to perform allosteric functions. Comparing directed aging [Eq. (14)] with coupled learning [Eq. (7)], we see that the clamped terms are the same, but the directed aging rule is missing the free term. As a special case of coupled learning, directed aging is expected to perform well in systems whose energy in the free state (or its derivative) can be neglected. Appendix B shows that in general both the free and clamped terms are necessary to train

networks for desired functions. Therefore, coupled learning can be viewed as a generalization of the directed aging framework that is successful for a more general class of physical networks.

V. DISCUSSION

In this work we have introduced coupled learning, a class of learning rules born of contrastive Hebbian learning and equilibrium propagation [15,18], and applied it to two types of physical systems, namely flow networks and mechanical spring networks. The advantage of such supervised learning rules, compared to more traditional techniques such as optimization of a cost function, is that they are physically plausible; at least in principle, coupled learning can be implemented in realistic materials and networks, allowing them to learn autonomously from external signals.

Such learning machines are not only interesting in themselves, but may have important advantages compared to physical systems designed by optimizing a cost function. First, because the process involves local responses to local stresses, the approach is scalable—training steps in networks of different sizes (different numbers of nodes) would take approximately the same amount of time. In contrast, the time required to compute gradients of a collective cost function increases rapidly with system size.

Second, the ability to train the system *in situ* means that it is not necessary to know the network geometry or topology, or even any information about the physical or learning degrees of freedom. This is particularly valuable for experimental systems, which do not have to be characterized in full microscopic detail in order to be trained, as long as the proper learning rules can be implemented, at least in an approximate form.

Third, as long as the learning rules can be implemented, one does not need to manipulate individual edges to have the desired values of the learning degree of freedom (e.g., the edge conductance for a flow network). Thus, the role of the supervisor can be filled by an end user, training the network for their desired tasks, rather than an expert designer.

An experimental realization of a learning flow network seems quite plausible, as has also been suggested for analog resistor networks [22]. The required ingredients are pipes whose conductances can be modified in response to the current carried by the pipe. It is possible that this condition is similar to that used by the brain vasculature, where vessels can be expanded or constricted [42,43].

We have focused primarily on training physical networks for one particular function (i.e., one particular source-target map, namely, allostery). However, coupled learning rules may be used as a stochastic gradient descent step, training the network for a different function in each training iteration. This idea allowed us to train the flow network to distinguish MNIST digits. The dynamics of learning multiple functions using coupled learning is quite involved, and the training performance may depend strongly on the

order and frequency of shown training examples. We will address these issues in subsequent work.

One might ask whether physical networks could compete as learning machines with computational neural networks. Our aim is not to outperform neural networks. Rather, the goal is to design physical systems capable of adapting autonomously to tasks and able to generalize to diverse inputs without a computer. Nevertheless, physical networks do supply at least one potential advantage compared to computational neural networks. In contrast to feed-forward neural networks often used in machine learning, the input-output relations in our physical (recurrent) networks are necessarily symmetric in the linear response regime (the regime in which the target response is proportional to the source). As a result, training target responses for given sources may yield a *generative model* [44,45]. Such generative models could produce examples of a class given its label by imposing target values distributed around the trained responses, and reading out the free source values. We leave the prospects of physical learning of generative models to future study.

ACKNOWLEDGMENTS

APPENDIX A: NUDGE AMPLITUDE η

In the main text we used local learning rules based on the difference between the free state and a slightly nudged clamped state ($\eta \ll 1$). We showed how choosing $\eta \ll 1$ in flow networks allows the learning rule to mimic the optimization of a global cost function [Fig. 7(c)], and we discuss that further in Appendix B.

Here we argue that choosing large nudge amplitudes $\eta \sim 1$, as suggested by contrastive Hebbian learning [15], can adversely affect learning in physical networks, particularly in nonlinear networks (e.g., mechanical networks).

The choice $\eta \ll 1$ implies that the clamped state is almost identical to the free state. Inspecting Eq. (4) or Eq. (7), we see that the learning rule essentially becomes a derivative of the energy function with respect to the physical variables, in the direction toward the desired state.

Thus, the learning rule is in fact a derivative of the energy in both spaces, those of the physical (e.g., pressure values) and learning degrees of freedom (e.g., pipe conductances). The choice $\eta \ll 1$ implies a local modification of the system not only in the spatial sense (so that only nearby elements communicate), but also in the generalized energy landscape of the combined configuration space of physical and learning parameters.

Conversely, choosing a large nudge amplitude $\eta \sim 1$ means the free and clamped states can be far away from each other, so that the learning rule does not approximate the derivative very well. This may be particularly important in nonlinear networks whose energy landscape is nonconvex. In such cases, if the free and clamped states are far apart, they may belong to different attractor basins in the landscape, possibly impeding the learning process, since it is not guaranteed that the learning rule can eliminate energy barriers between the two states. This problem has been long recognized as a limitation of contrastive Hebbian learning [15], and has been solved by using a small nudging factor [18].

We thus argue for the benefit of choosing small nudges $\eta \ll 1$. To test this proposal, we observe the learning process in flow networks and mechanical networks with different choices of η (Fig. 8). We trained flow networks
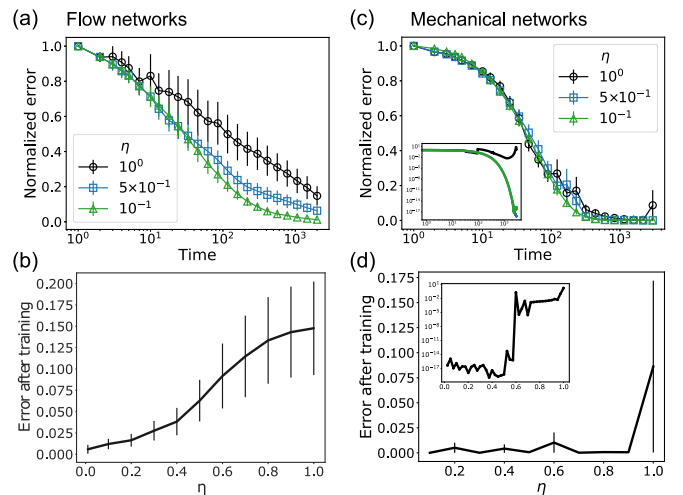

FIG. 8. Effects of nudge amplitude η on learning. (a) 512 node flow networks trained with 10 random source nodes and 3 target nodes. As the nudge amplitude η is increased, training gradually and continuously becomes slower. (b) Trained flow networks (after 2×10^3 time steps) perform better if trained with small nudges. (c) When learning is attempted in nonlinear mechanical networks using the ℓ rule, a similar picture emerges *on average*, where learning degrades at higher η. Moreover, at particular realizations, we observe noncontinuous degradation of the learning process at particular values of η (inset). (d) On average, learning performance is largely maintained until $\eta \sim 1$. In one 512 node network, trained with 10 random sources and 3 random targets, training is relatively stable until $\eta \approx 0.6$, and then goes thorough a discontinuous jump (inset). Averaged data results from 17 realizations of learning problems.

(with 512 nodes, 10 sources, and 3 targets) with varying values of the nudge parameter η [Figs. 8(a) and 8(b)]. It is generally found that choosing small nudge values $\eta \ll 1$ leads to better learning performance. The learning process is faster for lower η, so that after a fixed training time, the accuracy of networks trained with small η is better. When η is raised gradually, we find that the learning process is gradually and continuously slowed down.

Changing the nudge amplitude for nonlinear networks, such as a mechanical spring network [Figs. 8(c) and 8(d)] can lead to qualitatively different results. In one network of 512 nodes [trained with the ℓ-learning rule of Eq. (7), 10 random sources and 3 random targets], we find a discontinuous behavior at $\eta \approx 0.6$. At lower values we observe similar behavior to a linear flow network, but at higher η learning is abruptly slowed down by orders of magnitude. As discussed above, this effect likely stems from the issue that for this learning problem, the initial free state and the desired states belong to different attractor basins in the network energy landscape. Thus, when η is large enough, the free and clamped states belong to different basins, significantly slowing down the learning process. That being said, these nonlinear networks can still learn with large nudges $\eta \sim 1$, reaching low-error values.

We conclude that choosing small nudge values $\eta \ll 1$ yields superior results when training our networks. Such choice leads to consistent and continuous learning in both linear and nonlinear networks, as the learning rule best approximates the gradient descent on the desired cost function (see Appendix B). When choosing large nudges $\eta \sim 1$, learning efficiency usually deteriorates, and we observe interesting dynamical effects such as intermittent unlearning events. We will address the dynamics of large nudges in future work.

APPENDIX B: EFFECTIVE COST FUNCTION

In the main text we established that the coupled learning rule trains networks to present desired source-target relations. Surprisingly, the local learning rule predicted by our framework is very similar to the global modification resulting from minimizing a cost function. This result is very encouraging, as it shows that physically plausible, local learning rules can give rise to comparable training as nonlocal gradient descent, which is the ultimate greedy training strategy. In this Appendix, we explain this surprising similarity by deriving an effective cost function from our learning rule.

As discussed in the main text, the network autonomously modifies its edge parameters w by subtracting the energies of free and clamped states and taking a derivative:

$$\dot{w} \sim \partial_w \{ E^F(x_S, x_T^F, x_H^F) \\ - E^C(x_S, x_T^F + \eta[X_T - x_T^F], x_H^C) \}. \quad (B1)$$

Here, we expanded the notation of the main text to include the free and clamped states of all the hidden nodes x_H. Given a small nudge parameter $\eta \ll 1$, the clamped state is very similar to the free state, and we can approximate the clamped state using a Taylor expansion around the free state. In particular, let us write the clamped state of the hidden nodes as $x_H^C \equiv x_H^F + \eta \Delta x_H$, with the shift $\eta \Delta x_H$ a vector of magnitude $\sim \eta$. Further, we define the signed error in the target signal $\epsilon_T \equiv X_T - x_T^F$. Concatenating the source, target, and hidden nodes together, the difference between the free and clamped state can be written as

$$\Delta x \equiv (x_S^C - x_S^F, x_T^C - x_T^F, x_H^C - x_H^F) = \eta(0, \epsilon_T, \Delta x_H).$$

Now, we can approximate the energy of the clamped state E^C as

$$E^C \approx E^F + \eta(0, \epsilon_T, \Delta x_H)\nabla_x E^F \\ + \frac{1}{2}\eta^2 (0, \epsilon_T, \Delta x_H)^T \frac{\partial^2 E^F}{\partial x \partial x}(0, \epsilon_T, \Delta x_H). \quad (B2)$$

By definition of the free state, it is a stable equilibrium state of the physical network, so that the first-order term vanishes, $\nabla_x E^F = 0$. The second derivative in the second-order term is the Hessian at the free state H^F. Thus the Taylor expansion simplifies to

$$E^C \approx E^F + \frac{\eta^2}{2}(0, \epsilon_T, \Delta x_H)^T H^F (0, \epsilon_T, \Delta x_H). \quad (B3)$$

This result can be plugged back into the learning rule of Eq. (B1) to obtain

$$\dot{w} \sim -\frac{\eta^2}{2} \partial_w (0, \epsilon_T, \Delta x_H)^T H^F (0, \epsilon_T, \Delta x_H). \quad (B4)$$

As discussed in the main text, a typical tuning algorithm starts by defining a cost function $\mathcal{C} \equiv \epsilon_T^2 = \epsilon_T^T I \epsilon_T$, with I the identity matrix. This cost function is minimized via gradient descent:

$$\dot{w}_G \sim -\partial_w \mathcal{C}. \quad (B5)$$

Comparing Eqs. (B4) and (B5), we may identify an effective cost function minimized by our learning rule:

$$\mathcal{C} \sim \epsilon_T^T I \epsilon_T, \\ \mathcal{C}_{\text{eff}} \sim (0, \epsilon_T, \Delta x_H)^T H^F (0, \epsilon_T, \Delta x_H). \quad (B6)$$

While the effective cost function we derived is not identical to the standard cost function, they share important properties. First, both functions are convex (square forms) in the difference between the desired and the obtained result ϵ_T. Both functions are bounded from below by 0, as the Hessian of the stable free state is positive semidefinite. The two cost functions vanish if and only if the obtained

targets are equal to the desired targets $x_T^F = X_T$. Note that this is true since at $\epsilon_T = 0$, Δx_H vanishes as well. Overall, we know that a global minimum of $\mathcal{C}_{\text{eff}}$ must be a global minimum of $\mathcal{C}$. The effect of the Hessian matrix close to that minimum is mostly to stretch and rotate the paraboloid cup implied by the square form. When comparing the two cost functions, we also find that their gradients are aligned to significant extents as shown in Fig. 7. While this alignment is not guaranteed and is affected by many factors (e.g., network physics and architecture, number of targets, error values, etc.), the fact that both cost functions share the same minima means that coupled learning will tend to find solutions that minimize the original cost function too. We conclude that minimizing $\mathcal{C}_{\text{eff}}$ should mimic the minimization of $\mathcal{C}$, at least close enough to a minimum of $\mathcal{C}$, as is in fact observed.

Recently, it was shown [22,23] that using a nudged state that applies a "force" on the output nodes, rather than clamping their values as done here, implies that the effective cost function approaches the original cost function $\mathcal{C}$ for small nudge amplitudes $\eta \to 0$.

APPENDIX C: EXPERIMENTALLY FEASIBLE LEARNING RULE FOR FLOW NETWORKS

As discussed in the main text, implementing the full coupled learning rule of Eq. (4) in an experimental setting may prove difficult. One would need to implement pipes that can adapt in opposite ways to the same flows, depending on if the system is set in the free or clamped state. To simplify this learning rule for better experimental feasibility, we start by considering the full rule for conductance adaptation:

$$\dot{k} \sim \{[\Delta p^F(p_S, p_T^F)]^2 - [\Delta p^C(p_S, p_T^F + \eta[P_T - p_T^F])]^2\}. \quad (C1)$$

Recall that Δp is a vector of pressure differences across all pipes. It may be replaced by the difference of pressures in the two nodes connected by the pipe $\Delta p \equiv p_2 - p_1$. We can use this form in Eq. (C1) to obtain:

$$\dot{k} \sim \{[p_2^F - p_1^F]^2 - [p_2^C - p_1^C]^2\}$$
$$= (p_1^F - p_1^C)(p_1^F + p_1^C) + 2p_1^C p_2^C$$
$$+ (p_2^F - p_2^C)(p_2^F + p_2^C) - 2p_1^F p_2^F. \quad (C2)$$

Now, we define the difference between the free and clamped states as $\delta p_\mu \equiv p_\mu^F - p_\mu^C$, with μ the node index. If we choose a small nudge parameter $\eta \ll 1$, the free and clamped states are almost identical, such that $p_\mu^F \approx p_\mu^C \to |\delta p_\mu| \ll |p_\mu^F|$. Using the proximity of the free and clamped states, we can approximate Eq. (C2):

$$\dot{k} \sim 2[\delta p_1 p_1^F + \delta p_2 p_2^F - p_1^F p_2^F + (p_1^F - \delta p_1)(p_2^F - \delta p_2)]$$
$$\approx 2(\delta p_1 - \delta p_2)(p_1^F - p_2^F). \quad (C3)$$

Going back to the notation of pressure drops across pipes, we can write $\Delta p^\delta \equiv \delta p_1 - \delta p_2$, supporting the definition of the δ state as described in the main text. The approximated learning rule can thus be rewritten as

$$\dot{k} \approx 2\alpha\eta^{-1}\Delta p^F \Delta p^\delta. \quad (C4)$$

As long as we pick $\eta \ll 1$, this approximation is very accurate. To make this learning rule more experimentally feasible, we further approximate it so that only the sign of Δp^F is accounted for, and the conductances change in only one direction, yielding Eq. (8). Finally, we note that while this approximation is derived specifically for linear flow networks, the derivation may be similar for diverse physical networks, as long as the energy of an edge is (to first approximation) proportional to the difference in "activation" between the connected nodes. For example, similar approximation could be derived for both the k and ℓ learning rules in elastic networks [Eq. (7)].

APPENDIX D: NON-NEGATIVE LEARNING PARAMETERS

In this work we introduced coupled learning, a supervised learning framework for physical networks. Using coupled learning, one can derive the relevant learning rules (i.e., the proper dynamics for modifying the learning degrees of freedom). The systems demonstrated in this work, flow and elastic networks, are special examples. However, the learning degrees of freedom in both examples have a shared physical limitation: The conductance of pipes in a flow network, as well as the stiffness and rest lengths of springs in an elastic networks, are all defined as nonnegative quantities. This is a similar issue to the nonnegativity of synaptic connection strengths in natural neural networks [46].

Because of this physical limitation of the networks we considered, their capacity to learn arbitrary mappings between the sources and targets is limited. For example, while linear flow networks will always give rise to a linear mapping between the pressures at the sources and targets, the nonnegativity limitation excludes many conceivable linear mappings, depending on the geometry of the network and source pressures. Therefore, we expect that a linear flow network would generally be less successful than a general linear model (i.e., linear regression) in learning certain desired tasks.

While several possible approaches for circumventing this issue have been proposed [47,48], here we discuss the effect of such non-negativity limitation on the learning process in our physical networks. To avoid negative values in the learning degrees of freedom, we simply cut off their values at a small number (e.g., $k \geq 10^{-6}$ in flow networks).
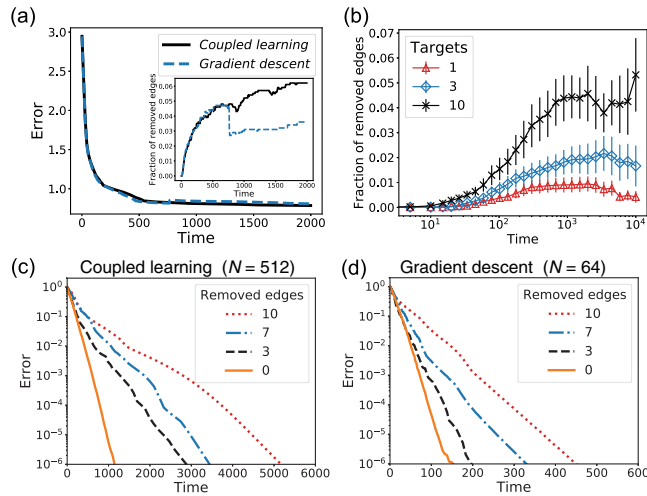
FIG. 9. Effects of non-negative learning degrees of freedom. (a) Training a flow network ($N = 512$, $M_S = 10$, $M_T = 10$) on a "hard" task where learning only reduces the error by a small factor. We find that when coupled learning fails to train a flow network, gradient descent (tuning) fails as well, suggesting the issue lies with the network and not the learning rule. Inset shows that a few percent of edges (conductances) reach their minimal value during training. (b) Averaging over 10 training tasks, we find that harder training problems with more simultaneous targets induce a larger fraction of edges to reach their minimal value. (c) Median normalized error during training with coupled learning for networks with different number of vanishing edges, showing that the non-negative constraint slows learning ($N = 512$, $M_S = 10$, $M_T = 5$). (d) A similar slowdown arises when the network is trained with standard gradient descent ($N = 64$, $M_S = 10$, $M_T = 5$).

We find that training the networks often causes a fraction of the learning parameters to reach these small cutoff values. This may significantly affect learning for two reasons. First, the magnitude of modification to the learning parameters is reduced, so that learning may fail or slow down. Second, the modification of the learning parameters becomes less aligned with the gradient of the cost function.

In Fig. 9, we again directly compare coupled learning and tuning (gradient descent) for training of flow networks, as shown in Fig. 7. However, in this example ($N = 512$, $M_S = 10$, $M_T = 10$), learning "fails," so that the initial error is only reduced by a factor of 3. It is important to note that the failure is shared by both coupled learning and gradient descent, hinting that the problem is not in the learning rule, but rather in the network physics or architecture. Indeed, the inset of Fig. 9(a) shows that as training proceeds, a few percent of the edges vanish, with their conductance values reaching the minimal cutoff. In Fig. 9(b), we average the fraction of vanishing edges over 10 training examples to show that harder training problems (with more simultaneous targets) often lead to a larger fraction of vanishing conductance values. One may ask whether this vanishing of edges causes problems for

training. In Figs. 9(c) and 9(d), we find that learning slows down considerably with an increasing number of removed edges for tasks with the same number of targets ($M_T = 5$), regardless of whether training is performed using our coupled learning approach or tuning (gradient descent). Note that the gradient descent case trains faster because we consider a much smaller $N = 64$ network [see Figs. 2(e) and 2(f)]. Together with Fig. 9(b), these results suggest that it is the physical constraint of non-negative learning degrees of freedom that slows down learning, not the coupled learning protocol itself.

[1] D. Norman, *The Design of Everyday Things*, Revised and Expanded Edition (Basic Books, New York, NY, 2013).

[2] S. H. Thomke, *Managing Experimentation in the Design of New Products*, Manage. Sci. **44**, 743 (1998).

[3] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, *Supervised Machine Learning: A Review of Classification Techniques*, Emerg. Artif. Intell. Appl. Comput. Eng. **160**, 3 (2007), https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf.

[4] 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).

[5] J. W. Rocks, N. Pashine, I. Bischofberger, C. P. Goodrich, A. J. Liu, and S. R. Nagel, *Designing Allostery-Inspired Response in Mechanical Networks*, Proc. Natl. Acad. Sci. U.S.A. **114**, 2520 (2017).

[6] J. W. Rocks, H. Ronellenfitsch, A. J. Liu, S. R. Nagel, and E. Katifori, *Limits of Multifunctionality in Tunable Networks*, Proc. Natl. Acad. Sci. U.S.A. **116**, 2506 (2019).

[7] B. A. Richards and T. P. Lillicrap, *Dendritic Solutions to the Credit Assignment Problem*, Curr. Opin. Neurobiol. **54**, 28 (2019).

[8] J. W. Rocks, A. J. Liu, and E. Katifori, *Revealing Structure-Function Relationships in Functional Flow Networks via Persistent Homology*, Phys. Rev. Research **2**, 033234 (2020).

[9] N. Pashine, D. Hexner, A. J. Liu, and S. R. Nagel, *Directed Aging, Memory, and Natures Greed*, Sci. Adv. **5**, eaax4215 (2019).

[10] 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. Research **2**, 043231 (2020).

[11] D. Hexner, A. J. Liu, and S. R. Nagel, *Periodic Training of Creeping Solids*, Proc. Natl. Acad. Sci. U.S.A. **117**, 31690 (2020).

[12] H. Ronellenfitsch and E. Katifori, *Global Optimization, Local Adaptation, and the Role of Growth in Distribution Networks*, Phys. Rev. Lett. **117**, 138301 (2016).

[13] 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**, 14843 (2020).

[14] M. Stern, V. Jayaram, and A. Murugan, *Shaping the Topology of Folding Pathways in Mechanical Systems*, Nat. Commun. **9,** 4303 (2018).

[15] J. R. Movellan, *Contrastive Hebbian Learning in the Continuous Hopfield Model*, *Connectionist Models* (Elsevier, Amsterdam, 1991), pp. 10–17.

[16] P. Baldi and P. Sadowski, *A Theory of Local Learning, the Learning Channel, and the Optimality of Backpropagation*, Neural Netw. **83,** 51 (2016).

[17] Y. Bengio and A. Fischer, *Early Inference in Energy-Based Models Approximates Back-Propagation*, arXiv:1510.02777.

[18] B. Scellier and Y. Bengio, *Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation*, Front. Comput. Neurosci. **11,** 24 (2017).

[19] S. Bartunov, A. Santoro, B. Richards, L. Marris, G. E. Hinton, and T. Lillicrap, *Assessing the Scalability of Biologically-Motivated Deep Learning Algorithms and Architectures*, in *Proceedings of the 32nd Conference on Advances in Neural Information Processing Systems, Montreal, 2018* (Curran Associates Inc., Red Hook, NY, 2018), pp. 9368–9378.

[20] J. Z. Kim, Z. Lu, S. H. Strogatz, and D. S. Bassett, *Conformational Control of Mechanical Networks*, Nat. Phys. **15,** 714 (2019).

[21] M. Ruiz-García, A. J. Liu, and E. Katifori, *Tuning and Jamming Reduced to Their Minima*, Phys. Rev. E **100,** 052608 (2019).

[22] J. Kendall, R. Pantone, K. Manickavasagam, Y. Bengio, and B. Scellier, *Training End-to-End Analog Neural Networks with Equilibrium Propagation*, arXiv:2006.01981.

[23] A. Laborieux, M. Ernoult, B. Scellier, Y. Bengio, J. Grollier, and D. Querlioz, *Scaling Equilibrium Propagation to Deep ConvNets by Drastically Reducing Its Gradient Estimator Bias*, arXiv:2006.03824.

[24] G. Thimm and E. Fiesler, *High-Order and Multilayer Perceptron Initialization*, IEEE Trans. Neural Networks **8,** 349 (1997).

[25] J. Y. F. Yam and T. W. S. Chow, *A Weight Initialization Method for Improving Training Speed in Feedforward Neural Network*, Neurocomputing;Variable Star Bulletin **30,** 219 (2000).

[26] 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,** 225501 (2015).

[27] N. Pashine, *Local Rules for Fabricating Allosteric Networks*, arXiv:2101.06793.

[28] L. Bottou, *Stochastic Learning*, *Lecture Notes in Computer Science* (Springer, Berlin, 2003), pp. 146–168.

[29] Y. LeCun, C. Cortes, and C. J. Burges, *The MNIST Database of Handwritten Digits*, http://yann.lecun.com/exdb/mnist, 2010.

[30] S. L. Soo and S. K. Tung, *Deposition and Entrainment in Pipe Flow of a Suspension*, Powder Technol. **6,** 283 (1972).

[31] W. S. J. Uijttewaal and R. V. A. Oliemans, *Particle Dispersion and Deposition in Direct Numerical and Large Eddy Simulations of Vertical Pipe Flows*, Phys. Fluids **8,** 2590 (1996).

[32] J. Jin, S. Chen, and J. Zhang, *UV Aging Behaviour of Ethylene-Vinyl Acetate Copolymers (EVA) with Different Vinyl Acetate Contents*, Polymer Degradation Stability **95,** 725 (2010).

[33] A. Boubakri, N. Haddar, K. Elleuch, and Y. Bienvenu, *Impact of Aging Conditions on Mechanical Properties of Thermoplastic Polyurethane*, Mater. Des. **31,** 4194 (2010).

[34] A. Holtmaat and K. Svoboda, *Experience-Dependent Structural Synaptic Plasticity in the Mammalian Brain*, Nat. Rev. Neurosci. **10,** 647 (2009).

[35] N. Yasumatsu, M. Matsuzaki, T. Miyazaki, J. Noguchi, and H. Kasai, *Principles of Long-Term Dynamics of Dendritic Spines*, J. Neurosci. **28,** 13592 (2008).

[36] D. D. Stettler, H. Yamahachi, W. Li, W. Denk, and C. D. Gilbert, *Axons and Synaptic Boutons Are Highly Dynamic in Adult Visual Cortex*, Neuron **49,** 877 (2006).

[37] D. Ratna and J. Karger-Kocsis, *Recent Advances in Shape Memory Polymers and Composites: A Review*, J. Mater. Sci. **43,** 254 (2008).

[38] I. K. Kuder, A. F. Arrieta, W. E. Raither, and P. Ermanni, *Variable Stiffness Material and Structural Concepts for Morphing Applications*, Prog. Aerosp. Sci. **63,** 33 (2013).

[39] M. Ernoult, J. Grollier, D. Querlioz, Y. Bengio, and B. Scellier, *Updates of Equilibrium Prop Match Gradients of Backprop through Time in an RNN with Static Input*, Adv. Neural Inf. Process. Syst. **32,** 7081 (2019), https://proceedings.neurips.cc/paper/2019/file/67974233917cea0e42a49a2fb7eb4cf4-Paper.pdf.

[40] S. Fancher and E. Katifori, *Tradeoffs between Energy Efficiency and Mechanical Response in Fluid Flow Networks*, arXiv:2102.13197.

[41] B. A. Richards, T. P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R. P. Costa, A. de Berker, S. Ganguli *et al.*, *A Deep Learning Framework for Neuroscience*, Nat. Neurosci. **22,** 1761 (2019).

[42] M. J. Cipolla, *The Cerebral Circulation*, *Integrated Systems Physiology: From Molecule to Function*, 1st ed. (Morgan & Claypool Life Sciences, San Rafael, CA, 2009), pp. 1–59.

[43] Yu.-R. Gao, S. E. Greene, and P. J. Drew, *Mechanical Restriction of Intracortical Vessel Dilation by Brain Tissue Sculpts the Hemodynamic Response*, NeuroImage **115,** 162 (2015).

[44] A. Y. Ng and M. I. Jordan, *On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes*, *Advances in Neural Information Processing Systems* (MIT press (Bradford Book), Cambridge, MA, 2002), pp. 841–848.

[45] T. Jebara, *Machine Learning: Discriminative and Generative* (Springer Science & Business Media, Berlin, 2012), Vol. 755.

[46] C. Parisien, C. H. Anderson, and C. Eliasmith, *Solving the Problem of Negative Synaptic Weights in Cortical Models*, Neural Comput. **20,** 1473 (2008).

[47] 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 *Proceedings of the 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), 2016* (IEEE, New York, 2016), pp. 1–6.

[48] Z. Wang, C. Li, W. Song, M. Rao, D. Belkin, Y. Li, P. Yan, H. Jiang, P. Lin, M. Hu *et al.*, *Reinforcement Learning with Analogue Memristor Arrays*, National electronics review **2,** 115 (2019).