Efficient Machine Learning in a Memristor Network

Monica Conte[1]

[1]*Soft Condensed Matter & Biophysics, Debye Institute for Nanomaterials Science, Utrecht University, Princetonplein 1, 3584 CC Utrecht, The Netherlands.*

(Dated: April 3, 2025)

Inspired by the brain's energy-efficient processing, we propose a physical implementation of artificial neural networks to surpass digital systems in speed and energy efficiency. We consider a network with nodes connected by conical microfluidic memristor channels filled with aqueous electrolyte [1]. These memristors exhibit tunable conductance, responsive to electrical, chemical, and mechanical stimuli, as well as channel geometry. Based on this adaptability, we develop a hybrid training algorithm that adjusts either channel geometry or external stimuli as weights. Physical relaxation replaces the feedforward pass, while a combined physical-numerical gradient descent drives the backward pass. This approach enables tasks such as input-output voltage mapping and linear regression.

I. INTRODUCTION

Artificial neural networks (ANNs) have become central to many technological advancements, achieving remarkable success in tasks ranging from image recognition to natural language processing. These networks typically operate in two phases: inference and backpropagation. The energy requirements of ANNs has grown significantly, negatively impacting the efficiency and scalability of ANNs[1].

While the inference energy consumption has been addressed by hardware accelerators, the backpropagation still remains a problem. As an alternative to ANN performed strictly in-silico, physical neural networks (PNN) have been introduced, where physical processes rather than mathematical operations are trained[]. In fact, physical processes are often faster and consume less energy than their digital counterpart.

Not only, but physical neural networks make the similarity with the human brain more evident, opening the doors to the development of an artificial brain. For this, we choose to work with networks whose constructing elements are memristors living in aqueous solution and that respond to electrical, chemical and mechanical stimuli at the same time.

Fluidic channels are emerging as a promising technology in the field of neuromorphic computing due to their ability to emulate synaptic functions. These nanoscale devices can store information through conductance changes. Their unique properties make them suitable for implementing artificial neural networks and other types of electrical networks.

In recent years, machine learning algorithms have increasingly been applied to memristor networks to train specific input-output relations. These algorithms typically modify the dynamics of the memristors to achieve desired computational tasks.

The paper is roganized as follows. Section 2 reviews related work on memristor networks, machine learning algorithms, and physical learning methods. Section 3 describes the methods used in our study, including the network model, machine learning algorithm, and the physical learning approach. In Section 4, we present our proposed method for coupling memristors with capacitors and discuss the necessary properties for effective training. Section 5 details our experimental setup and implementation. The results and analysis are provided in Section 6, followed by conclusions and suggestions for future work in Section 7.

II. MEMRISTOR NETWORK

The PNN we consider is an electrical network make of nodes connected by resistive elements. As resistive elements, we consider conical microfluidic channels that are schematically represented in Fig. 1. An azimuthally sym-

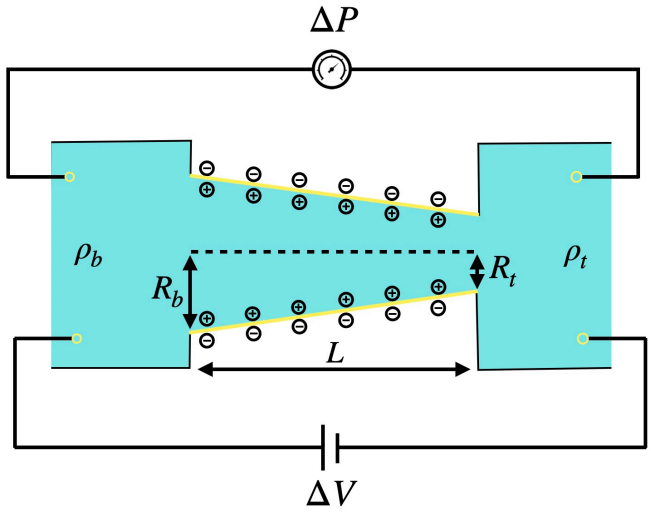

FIG. 1. A schematic representation of the conical fluidic memristor. A cone of lenght L, base radius R_b and tip radius R_t with charged surface connnects two bulk reservoirs of a 1:1 electrolyte in an acqueous solution. The bulk density in the base ρ_b differs from the bulk density in the tip ρ_t. At the far side of the memristor, an electrical potential and pressure drop are applied. Overall, this results in a chemical, electrical and mechanical sensistive system.

metric conical channel connects two reservoirs that contain an incompressible aqueous 1:1 electrolyte with viscosity $\eta = 1.01 mPas$ containing ions with diffusion coefficients $D = 1.75 \mu m^2 ms^{-1}$ and charge $\pm e$ with e the proton charge. The cone has a base radius of $R_b = 200$nm, a tip radius of $R_t = 50$nm and has a length $L = 10\mu$m, which sets it in the long channel limit where entrance and exit effects can be neglected. The walls of the channel are homogeneously charged and generate a surface potential ψ_0 that attracts the opposite-sign ions present in the solution due to the dissociation of the electrolyte. The electric potential decreases exponentially away from the surface due to the electrostatic screening of the attracted ions: an electric double layer (EDL) forms on the charged surfaces.

At the far side of both reservoirs, electrical potential, pressure and ion concentration are imposed. As a result, the memristor feels a potential drop $\Delta V = V_b - V_t$ defined as the difference between the potential in the base and in the tip. The ions within the EDL region generate an electro-osmotic fluid flow $Q_V = (\pi R_b R_t \epsilon \psi_0 / L\eta)\Delta V$. Meanwhile, the pressure drop $\Delta P = P_b - P_t$, given by the difference between the base and the tip pressure, induces a Poiseuille-like flow $Q_P = (\pi R_b^3 R_t^3 / 8 L \eta \langle R^2 \rangle)\Delta P$. Finally, the ions experience a concentration gradient $\Delta\rho = \rho_b - \rho_t$, defined by the difference between the ion concentration in the reservoir connected to the base of the channel, ρ_b, and the concentration at the tip, ρ_t. Consequently, a diffusive fluid flow arises. However, we consider only the pressure and potential contributions to the total fluid flow, $Q = Q_P + Q_V$, assuming the diffusive flow to be negligible. The resulting fluid flow generates non-trivial ion concentration profiles which, in turn, directly infulence the channel's conductance g.

In the steady-state, the channel is exposed to constant chemical, pressure and electrical signal, the corresponding conductance $g_\infty(\mathbf{\Delta})$ depends on the three stimuli $\mathbf{\Delta} = (\Delta V, \Delta P, \Delta\rho)$ and is determined by the ion concentration profiles according to

$$\frac{g_\infty(\mathbf{\Delta})}{g_0} = \int_0^L \frac{\bar{\rho}(x, \mathbf{\Delta})}{2\rho_b L} dx$$

where g_0 is the Ohmic conductance and $\bar{\rho}(x)$ the radially averaged salt concentration as a function of $x \in [0, L]$[2]. By solving the stationary condition of the total salt flux, an expression for the average salt concentration can be found and reads

$$\frac{\bar{\rho}(x, \mathbf{\Delta})}{2\rho_b} = 1 - \frac{\Delta\rho}{\rho_b}\mathcal{I}(x, \Delta V, \Delta P) + \frac{\rho_{in}(\Delta V)}{2\rho_b \mathrm{Pe}}\left[\mathcal{F}(x) - \mathcal{I}(x, \Delta V, \Delta P)\right] \quad (1)$$

where we defined, for the sake of simplicity and readability, two functions

$$\mathcal{I}(x, \Delta V, \Delta P) = \frac{e^{\frac{x}{L}\frac{R_t^2}{R_b R(x)}\mathrm{Pe}} - 1}{e^{\frac{R_t}{R_b}\mathrm{Pe}} - 1}, \qquad \mathcal{F}(x) = \frac{x}{L}\frac{R_t}{R(x)}.$$

Moreover, we introduced an expression for the concentration inhomogeneity

$$\rho_{in}(\Delta V) = 2\frac{(R_b - R_t)\sigma}{R_t^2}\frac{e\Delta V}{k_B T}$$

and the Péclet number $\mathrm{Pe} = QL/D\pi R_t^2$ which quantifies the importance of the flow Q over diffusion [3].

When the external stimuli vary dynamically in time, they generate ion accumulation and deplition in the channel, changing consequently the conductivity. We assume the dynamics of the time-dependent conductance $g(t)$ as a single exponential decay towards the steady-state conductance g_∞ with a diffusion-like timescale $\tau = L^2/12D$, as in Ref. [2]. The equation for the conductance reads

$$\frac{\partial g(\mathbf{\Delta}(t), t)}{\partial t} = \frac{g_\infty(\mathbf{\Delta}(t)) - g(\mathbf{\Delta}(t), t)}{\tau} \quad (2)$$

with $\mathbf{\Delta}(t) = (\Delta V(t), \Delta P(t), \Delta\rho(t))$. As in Ref. [2], we consider the time scale relative to voltage-driven ion accumulation or depletion in the channel. We justify this choice by stating that pressure-driven dynamics is faster

and recalling that we neglect chemically-driven transport. Due to the fact that the conductance depends on the history of the signals applied, this system is called memristor. Aside from the direct dependence of the conductance on the external stimuli, there is an implicit dependence on the geometrical properties such as lenght L and radius ratio R_b/R_t that we choose to consider.

The PNN is composed of N nodes, an example is shown in Fig. 2. Each node n $\in [1, \mathrm{N}]$ is characterized by a value

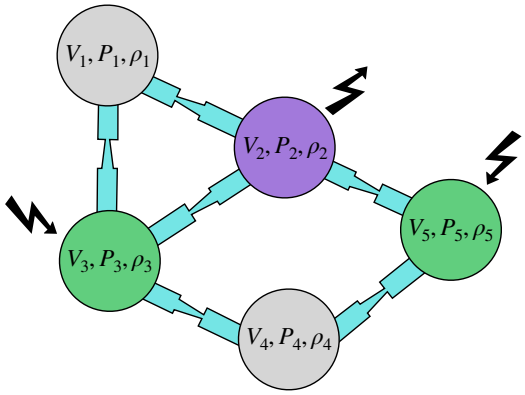

FIG. 2. An example of a network of five nodes connected by six memristors. The nodes are divided in input nodes (green circles), hidden nodes (gray nodes) and output nodes (purple circles) each of them is characterized by a triplet of physical quantities, representing the potential, pressure, and density. Th resulting stimuli on the memristor is given by the difference between base and tip signal, it thus depends on the oreintation of the memristor. In the case of the memristor that connects the first and the second node, for example, the exteranl stimuli are $\Delta V = V_2 - V_1$, $\Delta P = P_2 - P_1$ and $\Delta\rho = \rho_2 - \rho_1$. The zig-zag lines directed towards the input nodes indicate where the electrical signal is applied, while the zig-zag lines pointing outwards from the nodes indicate where the electrical signal is measured.

of electrical potential V_n, pressure P_n and bulk density ρ_n. Even though each node in the network is characterized by three physical properties, only electrical current flows through the network. This current propagates via the memristive elements, enabling modifications to the electric potential at other nodes. In contrast, pressure and density are externally imposed at each node and do not propagate through the network. The set of nodes is divided in thee sets. A node belongs to the set of *input nodes* if it is used to imput current in the circuit, its value of voltage is known and fixed. After the imputs are imposed, the electrical signal travels throught the network and the resulting output potential is read in the nodes belonging to *output nodes*. The remaining nodes, that do not function as input or output, are included in the set of *hidden nodes*.

The nodes in the PNN are connected by M memristive elements. Consequently, a memristor connecting two nodes experinces potential drop, pressure drop and concentration gradient given by the difference between the node where the base of the memristor is connected to, and the node to which the tip is linked. The memristor is characterized by its conductance state g_m with m $\in [1, \mathrm{M}]$, consequence of both the geomtery parameters and the external stimuli.

To solve numerically the resulting circuit, it is necessary to simultaneously address the circuit equations and the time evolution equation for the memristors' conductance. The circuit equations are handled using the open-source circuit solver AHKAB[], which computes the electric potentials at the nodes and the currents through the edges discretizing the differential equations in timesteps Δt. To incorporate the memristive elements, we approximate the time evolution of the conductances using a discretized form of Eq. (2), employing the Euler-forward method. Specifically, the conductance at a given timestep is calculated from its value at the previous timestep as:

$$g_\mathrm{m}\left(\mathbf{\Delta}(t), t\right) = g_\mathrm{m}\left(\mathbf{\Delta}(t), t - \Delta t\right) + \frac{g_{\mathrm{m},\infty}\left(\mathbf{\Delta}(t) - g_\mathrm{m}\left(\mathbf{\Delta}, t\right)\right)}{\tau}\Delta t. \quad (3)$$

In this work, we only apply constant voltage supplies to the input nodes of the circuit. Nonetheless, we notice that solving the dynamical equations is needed due to the dynamical change in the conductance of the memristors that, in turn, results in time-dependent potential drops across them.

III. TRAINING ALGORITHM FOR A MEMRISTOR NETWORK

As in a traditional algorithm for an ANN, the core mechanism of the training is to modify the weights associated to the connections of the network such that it maps specific inputs to desired outputs. In the case of the electrical network at hand, the training algorithm is designed to output desired voltage values in the output nodes, applying specific voltage values in the input nodes. Recalling that the dynamics of the memristors we consider depends on both external stimuli and geometrical properties, we have several possibilities for defining the weights of the network. However, there is a fundamental

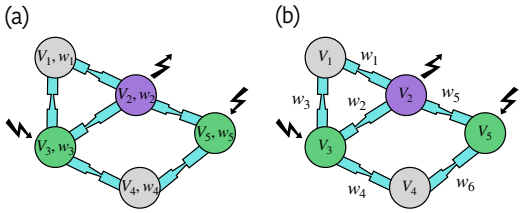

FIG. 3. (a) Memristor network with weights defined in the nodes. (b) Memristor network where wegiths characterize each edge.

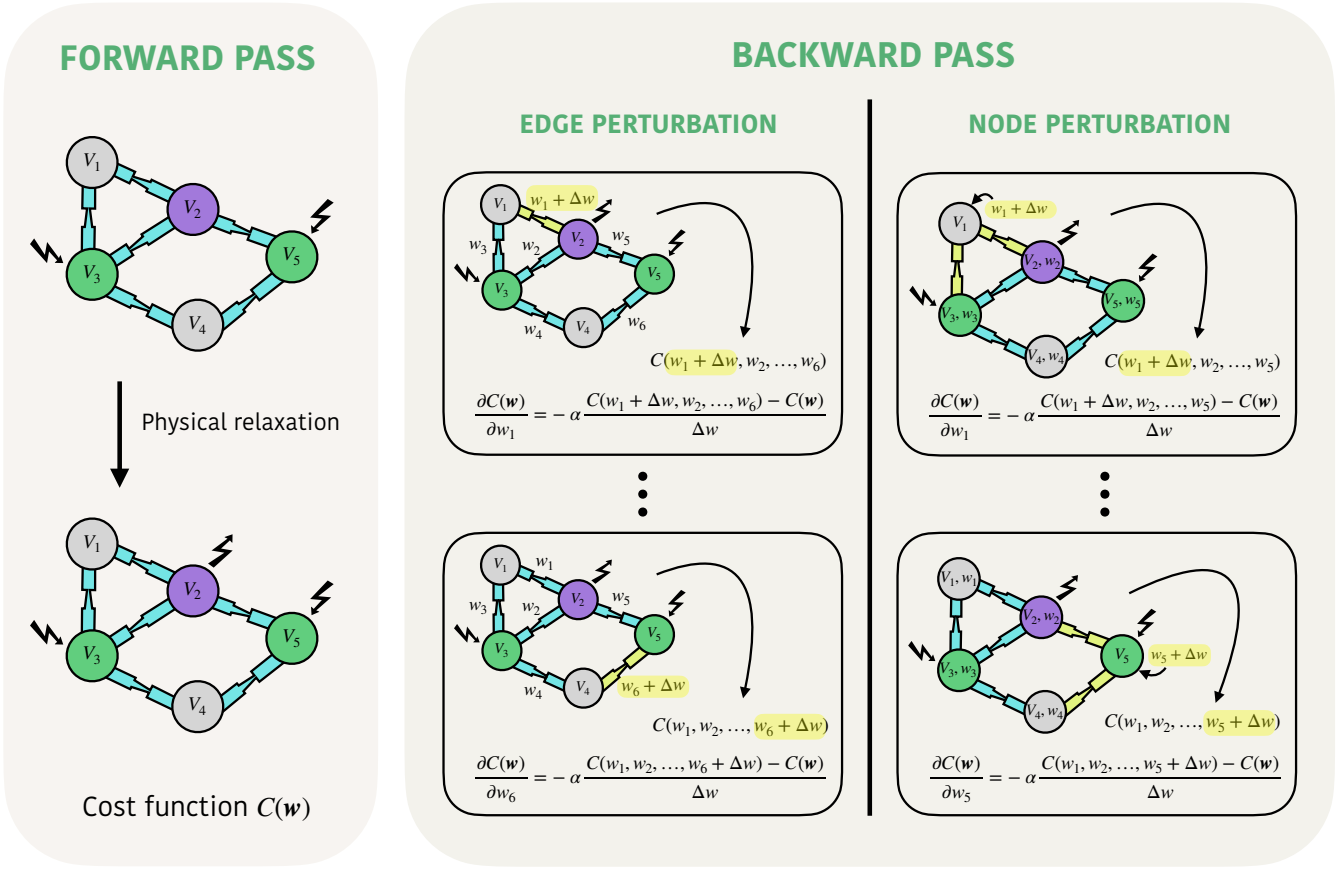

FIG. 4.

distinction between these two categories: external stimuli are applied to the nodes, while geometrical properties are defined edge-wise. Consequently, we distinguish between a training algorithm where the weights are defined on the nodes, as in Fig. 3(a), and one where the weights are defined on the edges, see Fig. 3(b). In this work, we focus on training a single property as the weight, while keeping all other parameters fixed as initially defined. For example, we may select an external stimulus, such as the imposed pressure or the density at each node. Alternatively, we can choose the length or radius ratio between the base and tip of the channel. The training process is carried out by iterating a scheme, illustrated in Fig. 4, which consists of two phases: a forward pass and a backward pass.

In the feedforward phase, the network is fed with specific electric potential in the input nodes. Following both circuit laws and memristor's dynamics, the network reaches a steady state where electrical potentials can be measured in the output nodes. Clearly, the output of the network depends on the set of weights imposed on the network $\boldsymbol{w}$, regardless them being node or edge defined. The value of the outputs can be used to quantify the status of the network, meaning that it indicated how far the answer of the network is from the desired one. Specifically, we define a cost function that computes the difference squared between the voltage in the output nodes and the values of desired voltages

$$C\left(\boldsymbol{w}\right) = \sum_{i \in \Lambda} \left(V_i\left(\boldsymbol{w}\right) - V_i^D\right)^2,$$

where Λ represents the set of indeces belonging to the output nodes, $V_i\left(\boldsymbol{w}\right)$ the voltage measured in the output nodes and V^D is a vector containing the values of desired output nodes.

The backward phase has the fundamental purpose of finding a set of weights that minimize the cost function. We follow the method of the *steepest descent*, for which the update of the weights from one training step s to the following $s+1$,

$$\boldsymbol{w}^{s+1} = \boldsymbol{w}^s - \alpha \boldsymbol{\nabla} C\left(\boldsymbol{w}\right), \tag{4}$$

results in a decrease in the cost funtion until a minimum is reached, for sufficienlty small values of α [4]. The gradient in the update rule is computed using a *finite differences method*

$$\boldsymbol{\nabla} C\left(\boldsymbol{w}\right) = \frac{C\left(\boldsymbol{w} + \boldsymbol{\Delta w}\right) - C\left(\boldsymbol{w}\right)}{\Delta w}, \tag{5}$$

where $\boldsymbol{\Delta w}$ is a vector with the same number of entries as the weight vector, but every entry has a value Δw.

In the training algortihm, we first initialize the system with the memristor parameters specified in Section II and with some initial pressure and concentration values at each node. With the knowledge of input and desired output voltage values, it is possible to performe the first forward pass, from which we obtain the cost function $C(\boldsymbol{w})$, that we store since it is needed in the computation its gradient. Subsequently, we proceed with the backward pass. We compute each entry of the gradient in Eq. (5) by selecting the corresponding weight, a node or an edge parameter, increment its value of Δw and applying a forward step to obtain the corresponding entry of $C(\boldsymbol{w} + \boldsymbol{\Delta w})$. With the stored value of $C(\boldsymbol{w})$, we are able to compute the gradient in Eq. (5) and use it to eventually calculate the weight update in Eq. (4).

Althought performed numerically in this work, we highlight that the feedforward pass is a physical process in which the system natually relaxes to its steady state. Moreover, also the backward pass we use can be translated to a physical event since involves performing the forward pass a number of time as the number of the weights. Once the cost functions are determined, only the simple algebric calculations of Eq. (4) are left to be computed. We peopse this method as a hybrid physical and in-silico training algorithm also as a future experimental approach to the probelm.

A. Training a Memristor Voltage Divider

A voltage divider is an electrical circuit made of three nodes connected in series by two resistances, that returns a fraction of the input voltage. Given the simplicity of its structure, we use it as an initial playground for the training algorithm described above. Specifically, we replace the resistances with memristor channels, to be trained such that giving $V_1 = 5V$ and $V_3 = 0V$ as inputs, we get $V_2 = 4V$ as output. In Fig. 5 a schematic of the memristor voltage divider is shown.

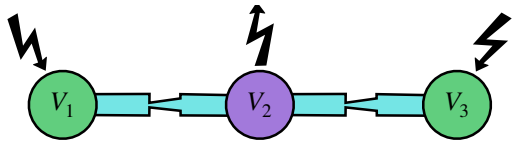

FIG. 5.

We initialize the system with pressures $P_1 = P_2 = 1bar$ and ion concetration $\rho_1 = \rho_2 = 0.2mM$ on each node, length $L_1 = L_2 = 10\mu m$ and base radius $R_{b1} = R_{b2} = 200nm$ on each edge. After choosing one of these four physical quantities as training parameter $\boldsymbol{w} = (w_1, w_2)$, we implement the training as descibed in Section III that modifies them, while the other parameters are kept constant. In Fig. 6 we show that the cost function normalized to its initial value decreases of many order of magnitude during the training for each of the choosen weight. The values of the choosen set of weights simultaneouly changes from the initial condition, as shown in Figures [], adjusting the behaviour of the network to the desired one. In fact, if the values of a choosen type of weights at the end of the training are imposed to the netowork together with the voltage inputs, the network will autonomously realx to a staedy state that coincides with the desired state. To exemplify this point, the inset in Figure shows the physical relaxation of the voltage divider when the trained values of pressure, which correspond to the three values in the last training step, are imposed on the nodes together with the constant inputs voltages. The inset shows that the output voltage V_2 autonomously relaxes to the desired value of $V_D = 3V$ in a time scale characteristic of the memristor.

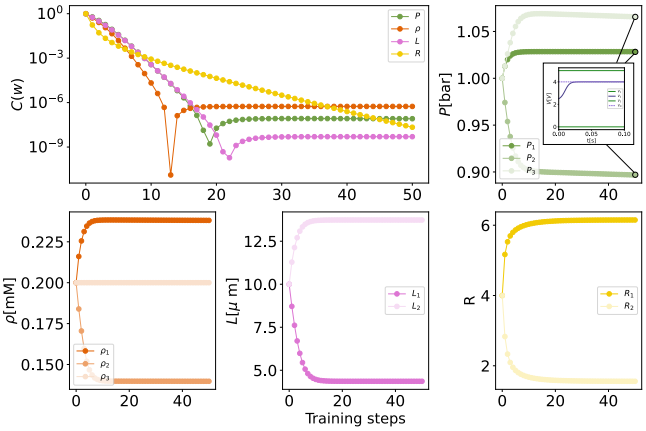

FIG. 6. (a) Behaviour of the cost function during the training of the voltage divider. Different colors indicate that the training is done with a specific choise of training weights, as indiacted in the legend. (b) Pressures imposed on the three nodes of the voltage divider during training. The inset shows the relaxation to the steady state of the output voltage V_2 to the desired value V_D when the pressures imposed are the values obtained in the last training step, and the input nodes are stimulated with fixed voltage. (c) Ion concentration desnities imposed on the three nodes of the voltage divider during training. (d) Length of the memristors during training. (e) Base radius of the memristors during training.

In order to investigate the adaptability of the system, we tested it with a set of three consequent desired voltages $V_D = \{1V, 3V, 4V\}$. In Fig. 7 we show how the output voltgage V_2 subsequently adapts to the desired voltages during the training of the pressures in the nodes, the evolution of which can be seen in Fig. 7. It is evident that the system can be retrained at any stage to a new desired objective making it reusable. Any other weight type can be chosen, depending on the necessity.

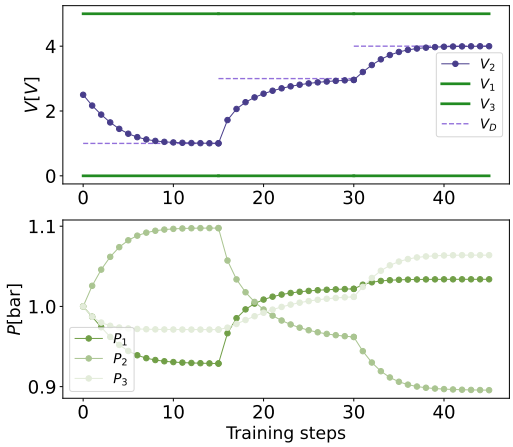

FIG. 7. (a) The trainiang of a memristor voltage divider to a set of desired voltage outputs $V_D = 1V, 3V, 4V$, indicated by dashed light purple lines. The output voltage V_2, indicated by dark purple dots, adapts to the desired output during the training. The input voltages V_1 and V_3 are constant at $5V$ and $0V$ respectively. (b) The dots with different shades of gray show the values of pressures, chosen as weights, imposed on the three nodes during training.

B. Training a memristor network

To generalize the training objective to a more general objective, we consider memrisotr networks with bigger networks with different topologies. Physically, reaching a desired input-output voltage mapping in a general electric circuit can be useful to adress desired voltage outputs in specific points of the circuit for variuous electronic purpuses.

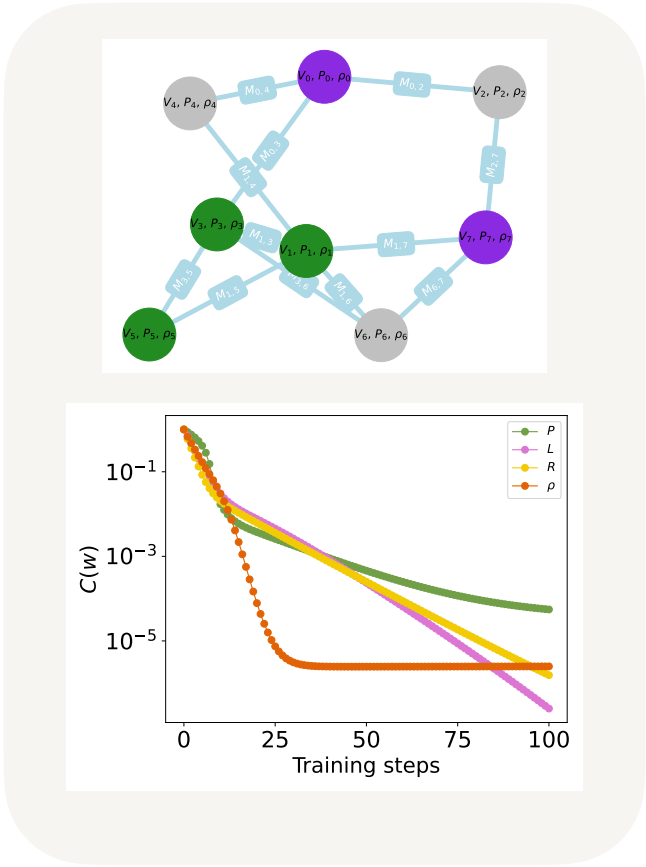

FIG. 8. Network to the right has inputs $[5, 0]$ and desired outputs $[3, 4]$, delta weights a third of their intial value and learning rates $1e^{-5}, 1e^{-4}, 1e^{2}$. The network to the right has $[5, 1, 0]$ and output $[3, 4, 2]$

C. Linear Regression

We broaden the training capability of the network by including a linear regression task, where, in its simplest form, the network is trained to give potential output V^{out} related to the input potential V^{in} via a linear relationship

$$V_{\text{out}} = \text{a}V_{\text{in}} + \text{b}. \tag{6}$$

We generate two sets of data, a training test $\{V_{\text{in}}, V_{\text{out}}\}_{i=1}^{T}$ and a test set $\{V_{\text{in}}, V_{\text{out}}\}_{i=1}^{Te}$. To create the test set, a random number in the interval $[1, 4]V$ is created and fed to equation Eq. (6)

IV. CONCLUSION

[1] Shih-Kai Chou, Jernej Hribar, Mihael Mohorčič, and Carolina Fortuna. The energy cost of artificial intelligence of things lifecycle, 2024.

[2] T. M. Kamsma, W. Q. Boon, T. ter Rele, C. Spitoni, and R. van Roij. Iontronic neuromorphic signaling with conical microfluidic memristors. *Phys. Rev. Lett.*, 130:268401, Jun 2023.

[3] Willem Q. Boon, Tim E. Veenstra, Marjolein Dijkstra, and René van Roij. Pressure-sensitive ion conduction in a conical channel: Optimal pressure and geometry. *Physics of Fluids*, 34(10), October 2022.

[4] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.