A Fast Algorithm to Simulate Nonlinear Resistive Networks

Benjamin Scellier [1]

Abstract

Analog electrical networks have long been investigated as energy-efficient computing platforms for machine learning, leveraging analog physics during inference. More recently, resistor networks have sparked particular interest due to their ability to learn using local rules (such as equilibrium propagation), enabling potentially important energy efficiency gains for training as well. Despite their potential advantage, the simulations of these resistor networks has been a significant bottleneck to assess their scalability, with current methods either being limited to linear networks or relying on realistic, yet slow circuit simulators like SPICE. Assuming ideal circuit elements, we introduce a novel approach for the simulation of nonlinear resistive networks, which we frame as a quadratic programming problem with linear inequality constraints, and which we solve using a fast, exact coordinate descent algorithm. Our simulation methodology significantly outperforms existing SPICE-based simulations, enabling the training of networks up to 327 times larger at speeds 160 times faster, resulting in a 50,000-fold improvement in the ratio of network size to epoch duration. Our approach can foster more rapid progress in the simulations of nonlinear analog electrical networks.

1. Introduction

As energy costs associated with machine learning are rapidly increasing, neuromorphic computing platforms are being explored as alternatives to neural networks and GPUs (Marković et al., 2020). Such platforms leverage analog physics and compute-in-memory architectures to achieve significant energy gains. In particular, nonlinear resistive networks have recently sparked interest (Kendall et al., 2020; Dillavou et al., 2022; 2023; Wycoff et al., 2022; Anisetti et al., 2024; Stern et al., 2022; 2024; Kiraz et al., 2022; Watfa et al., 2023; Oh et al., 2023). Central to these networks are variable resistors (e.g. memristors) which act as trainable weights, coupled with diodes that introduce nonlinearities, as well as voltage and current sources for input signals. The conductances of these variable resistors can be adjusted, allowing the network to be trained to achieve specific computational tasks. Notably, like traditional neural networks, nonlinear resistive networks have two essential properties: they are universal function approximators (Scellier & Mishra, 2023) and they can be trained by gradient descent (Kendall et al., 2020; Anisetti et al., 2024). Unlike GPU-based neural networks, however, they leverage the laws of electrical circuits (e.g. Kirchhoff's laws and Ohm's law) to perform inference and extract the weight gradients. Furthermore, the learning rules governing conductance changes are local. These features make nonlinear resistive networks good candidates as power-efficient learning-capable hardware, with recent experiments on memristive networks suggesting a potential 10,000x gain in energy efficiency compared to neural networks trained on GPUs (Yi et al., 2023). Small-scale self-learning variable resistor networks have been built and succesfully trained on datasets such as Iris (Dillavou et al., 2022; 2023), validating the soundness of the approach.

Simulations of larger nonlinear resistive networks on tasks such as MNIST (Kendall et al., 2020) and Fashion-MNIST (Watfa et al., 2023) further underscore their potential. These works and others (Kiraz et al., 2022; Oh et al., 2023) performed the simulations using the general-purpose SPICE circuit simulator (Keiter, 2014; Vogt et al., 2020). However, these efforts have been hampered by the slowness of SPICE, which is not specifically conceived to perform efficient simulations of resistive networks used for machine learning applications. To illustrate, Kendall et al. (2020) employed SPICE to simulate the training on MNIST of a one-hidden-layer network comprising 100 hidden nodes, a process which took one week for only ten epochs of training. Due to a scarcity of methods and algorithms for nonlinear resistive network simulations, another line of works resorts to linear networks (Stern et al., 2022; 2024; Wycoff et al., 2022), thus missing a fundamental ingredient of modern machine learning applications: nonlinearity. In order to better understand nonlinear resistive networks, guide their

[1]Rain AI, San Francisco, CA, USA. Correspondence to: Benjamin Scellier <benjamin@rain.ai>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

hardware design, and further demonstrate their scalability to more complex tasks, the ability to efficiently simulate them has thus become crucial.

We introduce a novel methodology tailored for the simulations of nonlinear resistive networks. Our algorithm, applicable to networks with arbitrary topologies, is an instance of an exact coordinate descent algorithm for convex quadratic programming (QP) problems with linear constraints (Wright, 2015). When applied to the 'deep resistive network' architecture of Kendall et al. (2020), our algorithm is an instance of an 'exact block coordinate descent' algorithm, whose runtime on GPUs is orders of magnitude faster than SPICE. The contributions of the present manuscript are the following:

- We show that, in a nonlinear resistive network, under an assumption of ideality of the circuit elements (resistors, diodes, voltage sources and current sources), the steady state configuration of node electrical potentials is the solution of a convex minimization problem: specifically, a convex quadratic programming (QP) problem with linear inequality constraints (Theorem 1).
- Using the QP formulation, we derive an algorithm to compute the steady state of an ideal nonlinear resistive network (Theorem 2). Our algorithm, which is an instance of an 'exact coordinate descent' algorithm, is applicable to networks with arbitrary topologies.
- For a specific class of nonlinear resistive networks called 'deep resistive networks' (DRNs), we derive a specialized, fast, algorithm to compute the steady state (Section 3). It exploits the bipartite structure of the DRN to perform exact block coordinate descent, where half of the coordinates (node electrical potentials) are updated in parallel at each step of the minimization process. Each step of our algorithm involves solely tensor multiplications, divisions, and clipping, making it amenable to fast executions on parallel computers such as GPUs.
- We perform simulations of ideal DRNs, trained with equilibrium propagation (EP) on the MNIST dataset (Section 4). Compared to the SPICE-based simulations of Kendall et al. (2020), our DRNs have up to 327 times more parameters (variable resistors) and the training time per epoch is 160 shorter, resulting in a 50000x larger network size to epoch duration ratio. We also train DRNs of two and three hidden layers for the first time.
- We compare in simulations DRNs with their closely related deep Hopfield networks (DHNs) studied in Scellier & Bengio (2017) (Appendix E). We show that, while both models are comparable in performance, DRNs require much fewer iterations to converge to steady state (e.g. 6 iterations for a 3-hidden-layer DRN vs 100 iterations for a 3-hidden-layer DHN).
- We prove that the equilibrium propagation formulas of Scellier & Bengio (2017); Scellier et al. (2024) remain valid in the setting of ideal nonlinear resistive networks, and more generally when the set of feasible configurations is defined by inequality constraints (Appendix F).

We emphasize that our methodology to simulate nonlinear resistive networks relies on an assumption of ideality of the circuit elements — an abstraction that might appear overly simplistic. Although the real-world deviations from ideality are acknowledged, the great acceleration in simulation times that our methodology offers opens the door to large-scale simulations of nonlinear resistive networks, which we believe may also be informative about the behaviour of real (non-ideal) networks.

2. Nonlinear Resistive Networks

In this work, we study electrical circuits composed of voltage sources, current sources, linear resistors and diodes. We refer to these circuits as nonlinear resistive networks.

2.1. Model and Assumptions

Following Scellier & Mishra (2023), we assume that the four circuit elements are *ideal*, with their behaviour determined by the following current-voltage (i-v) characteristics (Figure 1):

- Voltage Source: Satisfies $v = v_0$ for a constant voltage v_0, regardless of the current i.
- Current Source: Satisfies $i = i_0$ for a constant current i_0, regardless of the voltage v.
- Linear Resistor: Follows Ohm's law, $i = gv$, where g is the conductance ($g = 1/r$, with r being the resistance).
- Diode: Satisfies $i = 0$ for $v < 0$, and $v = 0$ for $i > 0$.

An ideal diode thus has two states: the off-state, where it behaves like an open switch, allowing no current to flow regardless of the voltage drop across it, or the on-state, where it behaves like a closed switch, allowing current to flow without any voltage drop.

A nonlinear resistive network can be represented as a graph where each branch contains a unique element, as shown in Figure 2. We denote the set of all branches as $\mathcal{B} = \mathcal{B}_{\rm VS} \cup \mathcal{B}_{\rm CS} \cup \mathcal{B}_{\rm R} \cup \mathcal{B}_{\rm D}$, where $\mathcal{B}_{\rm VS}$, $\mathcal{B}_{\rm CS}$, $\mathcal{B}_{\rm R}$ and $\mathcal{B}_{\rm D}$ are the subsets containing voltage sources, current sources, resistors

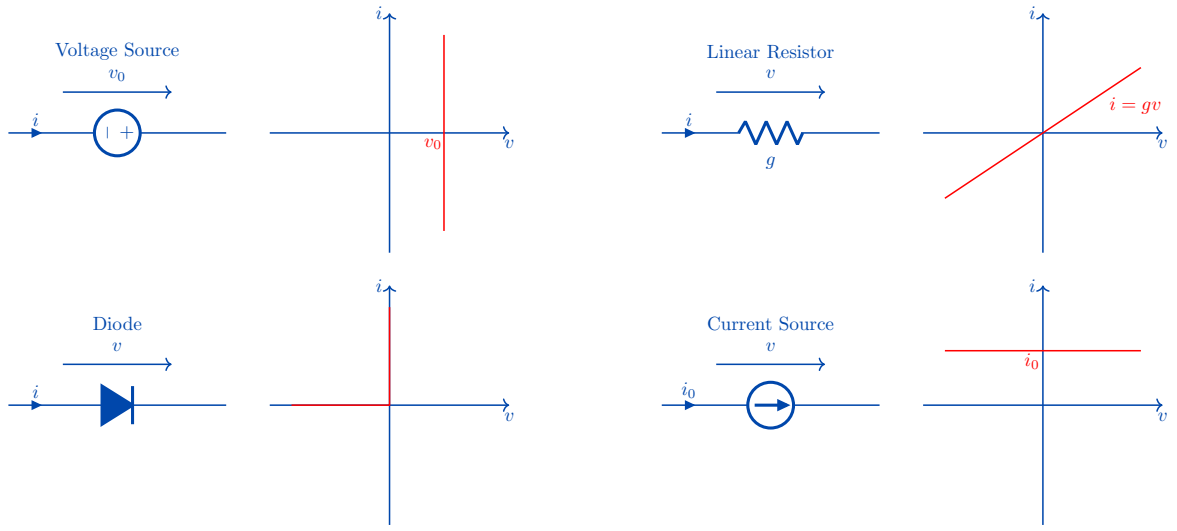

Figure 1. **Ideal circuit elements and their current-voltage (i-v) characteristics.** A linear resistor follows Ohm's law: $i = gv$, where g is the conductance ($g = 1/r$, with r being the resistance). An ideal diode is characterized by $i = 0$ for $v \leq 0$ ("off-state") and $v = 0$ for $i > 0$ ("on-state"). An ideal voltage source is characterized by $v = v_0$ for a constant voltage v_0 independent of the current i. An ideal current source is characterized by $i = i_0$ for a constant current i_0 independent of the voltage v.

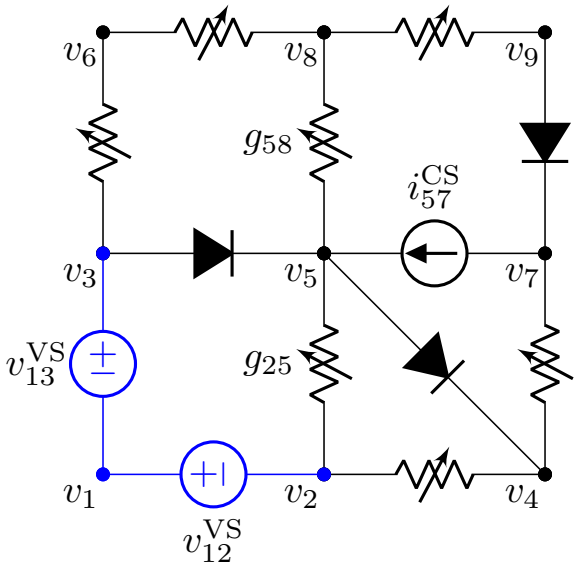

Figure 2. **A nonlinear resistive network.** By assumption, the voltage sources form a tree (in blue), so if we set e.g. $v_1 = 0$, we can immediately infer $v_2 = -v_{12}^{\mathrm{VS}}$ and $v_3 = v_{13}^{\mathrm{VS}}$. Next, we compute the steady state of the network by performing exact coordinate descent (Theorem 2) on the set of internal node electrical potentials (in black). As an example, one step of exact coordinate descent on node $k = 5$ proceeds as follows. First we look at the resistors and current sources connected to node $k = 5$ and we calculate $p_5 = (g_{25}v_2 + g_{58}v_8 + i_{57}^{\mathrm{CS}})/(g_{25} + g_{58})$. Then we look at the diodes connected to node $k = 5$ and we calculate $v_5 = \max(v_3, \min(p_5, v_4))$. This is the value of v_5 that achieves the minimum of $E(v_5)$ given other variables (node electrical potentials) fixed. We repeat the process with other nodes until convergence.

and diodes, respectively. For every $(j,k) \in \mathcal{B}_{\mathrm{VS}}$, we denote the voltage across the voltage source between nodes j and k as v_{jk}^{VS}. For every $(j,k) \in \mathcal{B}_{\mathrm{CS}}$, we denote the current through the current source between nodes j and k as i_{jk}^{CS}. Finally, for every $(j,k) \in \mathcal{B}_{\mathrm{R}}$, we denote the conductance of the resistor between nodes j and k as g_{jk}.

We define a 'steady state' as a configuration of branch voltages and branch currents that satisfies the above branch equations, as well as Kirchhoff's current law (KCL) at every node, and Kirchhoff's voltage law (KVL) in every loop. Under the above assumption of ideality, the steady state of a nonlinear resistive network is characterized by the following result – see Appendix A for a proof.

Theorem 1 (Convex QP formulation)**.** *Consider a nonlinear resistive network with N nodes, and denote $v = (v_1, v_2, \ldots, v_N)$ the vector of node electrical potentials*[1]*. Under the assumption of ideality, the steady state configuration of node electrical potentials, denoted $v_\star$, satisfies*

$$v_\star = \operatorname*{arg\,min}_{v \in \mathcal{S}} E(v), \tag{1}$$

where $E : \mathbb{R}^N \to \mathbb{R}$ is defined by

$$E(v_1, \ldots, v_N) := \frac{1}{2} \sum_{(j,k) \in \mathcal{B}_{\mathrm{R}}} g_{jk} \left(v_j - v_k\right)^2 \tag{2}$$

$$+ \sum_{(j,k) \in \mathcal{B}_{\mathrm{CS}}} i_{jk}^{\mathrm{CS}} \left(v_j - v_k\right), \tag{3}$$

and $\mathcal{S}$ is defined as:

$$\mathcal{S} := \{(v_1, v_2, \ldots, v_N) \in \mathbb{R}^N, \tag{4}$$

$$v_j \le v_k \quad \forall (j,k) \in \mathcal{B}_{\mathrm{D}}, \tag{5}$$

$$v_j = v_k + v_{jk}^{\mathrm{VS}} \quad \forall (j,k) \in \mathcal{B}_{\mathrm{VS}}\}. \tag{6}$$

A few comments are in order. First, $\mathcal{S}$ is the set of feasible configurations of node electrical potentials (or the *feasible set* for short), and E is the energy function of the network (or the *objective function*). Importantly, a configuration $v \in \mathcal{S}$ does not necessarily satisfy all the laws of electrical circuit theory - in other words, a *feasible* configuration is not necessarily the *physically realized* configuration. Similarly, the energy function E is defined for every feasible configuration $v \in \mathcal{S}$, even those that do not comply with all the laws of electrical circuit theory. Theorem 1 states that among all feasible configurations, the one that is physically realized (the steady state) is the configuration that minimizes E.

Second, the feasible set $\mathcal{S}$ is defined by linear equality and inequality constraints. The constraint $v_j \ge v_k$ for every $(j,k) \in \mathcal{B}_{\mathrm{D}}$ ensures the voltage $v_j - v_k$ across the diode is non-negative ($v_j > v_k$ if the diode is in the off-state, and $v_j = v_k$ if the diode is in the on-state). The constraint $v_j = v_k + v_{jk}^{\mathrm{VS}}$ for every $(j,k) \in \mathcal{B}_{\mathrm{VS}}$ ensures the voltage $v_j - v_k$ across the voltage source is v_{jk}^{VS}. The diodes and voltage sources thus constrain the set of feasible configurations. We note that some conditions on the network topology and branch characteristics must be met to ensure that the feasible set $\mathcal{S}$ is non-empty: for instance, if the network contains a loop of voltage sources whose voltage drops do not sum to zero, KVL is violated and the feasible set is empty.

Third, the energy function E is half the total power dissipated in the resistors plus the total power dissipated in the current sources. Specifically, $g_{jk}\left(v_j - v_k\right)^2$ represents the power dissipated in the resistor of branch $(j,k) \in \mathcal{B}_{\mathrm{R}}$, and $i_{jk}^{\mathrm{CS}}\left(v_j - v_k\right)$ represents the power dissipated in the current source of branch $(j,k) \in \mathcal{B}_{\mathrm{CS}}$. Theorem 1 generalizes the *principle of minimum dissipated power* which states that, in a linear resistor network (with voltage sources and linear resistors, but without diodes and current sources), among all feasible configurations of node electrical potentials, the one physically realized minimizes the power dissipated in the resistors. Other related results include Onsager's principle (Onsager, 1931) and Millar's theory (Millar, 1951), which provides a variational formulation for the steady state in a network composed of elements with arbitrary current-voltage (i-v) characteristics.

Finally, as a sum of convex functions, the energy $E(v)$ is a convex function of the node electrical potentials v. Specifically, the energy function $E(v)$ is a quadratic form in v, the Hessian of E being positive definite. The feasible set $\mathcal{S}$, defined by linear constraints, is also convex. Therefore, Theorem 1 states that the steady state configuration of node electrical potentials is the solution of a convex optimization problem, specifically, a convex quadratic programming (QP) problem with linear constraints.

Next we turn to our algorithm for simulating (ideal) nonlinear resistive networks.

2.2. An Algorithm to Simulate Nonlinear Resistive Networks

Equiped with the QP formulation (Theorem 1), we introduce a numerical method to compute the steady state $v_\star$ of an ideal nonlinear resistive network (1). Starting from some configuration $v^{(0)} \in \mathcal{S}$, our algorithm minimizes the energy function $E(v)$ with respect to $v \in \mathcal{S}$ by building a sequence of configurations $v^{(1)}, v^{(2)}, \ldots, v^{(t)}, \ldots$ in the feasible set $\mathcal{S}$ such that $E(v^{(t+1)}) \le E(v^{(t)})$ for every $t \ge 0$. Specifically, we use an 'exact coordinate descent' algorithm. The main ingredient of our algorithm is the result below.

For simplicity and clarity, we assume here that the voltage sources form a connected component – a tree of the network

[1]The electrical potentials are defined up to a constant, so we may assume, for instance, $v_1 = 0$.

– although our algorithm can be adapted to the more general case where the voltage sources form a forest (i.e a disjoint union of multiple trees).

Theorem 2 (Exact coordinate descent)**.** *Let $v = (v_1, \dots, v_N) \in \mathcal{S}$. Let k be an internal node of the network, i.e. a node that does not belong to the connected component of voltage sources. Define*

$$p_k := \frac{\sum_{j \in \mathcal{B}_\mathrm{R}} g_{kj} v_j + \sum_{j \in \mathcal{B}_\mathrm{CS}} i_{jk}^\mathrm{CS}}{\sum_{j \in \mathcal{B}_\mathrm{R}} g_{kj}} \tag{7}$$

and

$$v_k^\mathrm{min} := \max_{j:(j,k) \in \mathcal{B}_\mathrm{D}} v_j, \qquad v_k^\mathrm{max} := \min_{j:(k,j) \in \mathcal{B}_\mathrm{D}} v_j, \tag{8}$$

$$v_k' := \min\left(\max\left(v_k^\mathrm{min}, p_k\right), v_k^\mathrm{max}\right), \tag{9}$$

$$v' := (v_1, \dots, v_{k-1}, v_k', v_{k+1}, \dots, v_N). \tag{10}$$

Then, among all configurations $v'' \in \mathcal{S}$ of the form $v'' = (v_1, \dots, v_{k-1}, v_k'', v_{k+1}, \dots, v_N)$, the configuration v' is the one with the lowest energy, i.e.

$$v_k' = \operatorname*{arg\,min}_{v_k'' \in \mathcal{S}} E(v_1, \dots, v_{k-1}, v_k'', v_{k+1}, \dots, v_N). \tag{11}$$

In particular $v' \in \mathcal{S}$ and $E(v') \leq E(v)$.

We prove Theorem 2 in Appendix A. Below, we provide an intuitive explanation of the result. Let k such that $1 \leq k \leq N$. The energy function E of Eq. (2) is a quadratic function of v_k given the state of other node electrical potentials fixed $(v_1, \dots, v_{k-1}, v_{k+1}, \dots, v_N)$. In other words, E is of the form $E = a_k v_k^2 + b_k v_k + c_k$ for some real-valued coefficients a_k, b_k and c_k that do not depend on v_k. Specifically, the values of a_k and b_k are $a_k := \frac{1}{2} \sum_{j \in \mathcal{B}_\mathrm{R}} g_{jk}$ and $b_k := -\sum_{j \in \mathcal{B}_\mathrm{R}} g_{kj} v_j - \sum_{j \in \mathcal{B}_\mathrm{CS}} i_{jk}^\mathrm{CS}$. Furthermore, the range of feasible values for v_k, constrained by the diodes, is of the form $[v_k^\mathrm{min}, v_k^\mathrm{max}]$ where v_k^min and v_k^max are given by Eq. (8). Since the coefficient a_k is positive (as a sum of conductances), $E(v_k)$ is bounded below and its minimum in $\mathbb{R}$ is obtained at $p_k := -b_k/2a_k$. The minimum of $E(v_k)$ in the interval $[v_k^\mathrm{min}, v_k^\mathrm{max}]$ is found by clipping p_k between v_k^min and v_k^max. These boundaries reflect the constraints imposed by the diodes connected to node k, introducing nonlinearities.

Using Theorem 2, we can minimize the energy function E via an 'exact coordinate descent' strategy. Assuming that the feasible set $\mathcal{S}$ is not empty, we start from some feasible configuration $v \in \mathcal{S}$. Then, at each step, we pick some internal node k, and compute the value v_k' that minimizes E given the values of other variables fixed, using Theorem 2. This yields a new feasible configuration $v' \in \mathcal{S}$. We then pick another variable and repeat the process. At each step, the energy E either remains constant or decreases.

3. Deep Resistive Networks

We now turn to the deep resistive network (DRN) model, a layered nonlinear resistive network architecture introduced in Kendall et al. (2020). DRNs, which take inspiration from the architecture of layered neural networks, offer advantages for hardware design as they are potentially amenable for implementation using crossbar arrays of memristors (Xia & Yang, 2019). In this section, we show that ideal DRNs also present advantages in terms of simulation speed, when our algorithm is executed on multi-processor computers such as GPUs.

3.1. Deep Resistive Network Architecture

The DRN architecture is defined as follows. First of all, we choose a reference node called 'ground'. A set of nodes connected to ground by voltage sources form the 'input layer'. These voltage sources can be set to input values, playing the role of input variables. Another set of nodes form the 'output layer' whose electrical potentials play the role of model outputs.

Energy function. In a DRN, the circuit elements (voltage sources, resistors, diodes and current sources) are assembled into a layered network, mimicking the architecture of a deep neural network (Figure 3). For each ℓ such that $0 \leq \ell \leq L$, we denote N_ℓ the number of nodes in layer ℓ, where L is the number of layers in the DRN. Each node in the network is also called a 'unit' by analogy with a neural network. We denote $v_k^{(\ell)}$ the electrical potential of the k-th node of layer ℓ, which we may think of as the unit's activation. Pairs of nodes from two consecutive layers are interconnected by variable resistors - the 'trainable weights'. Denoting $g_{jk}^{(\ell)}$ the conductance of the variable resistor between the j-th node of layer $\ell - 1$ and the k-th node of layer ℓ, the energy function (2) of a DRN takes the form

$$E(v) = \frac{1}{2} \sum_{\ell=1}^{L} \sum_{j=1}^{N_{\ell-1}} \sum_{k=1}^{N_\ell} g_{jk}^{(\ell)} \left(v_j^{(\ell-1)} - v_k^{(\ell)}\right)^2. \tag{12}$$

Feasible set. The voltage sources and resistors being linear elements, the network built thus far is linear. To make it nonlinear, we use diodes. For each unit, we place a diode between the unit's node and ground, which can be oriented in either of the two directions. If the diode points from ground to the unit's node, the unit's electrical potential is non-negative: we call it an 'excitatory unit'. Conversely, if the diode points from the unit's node to ground, the unit's electrical potential is non-positive: we call it an 'inhibitory unit'. For each internal ('hidden') layer ℓ of the DRN $(1 \leq \ell \leq L - 1)$, we orient the diodes so that the units of even indices are excitatory and the units of odd indices are inhibitory. Finally, the units of the output layer $(\ell = L)$

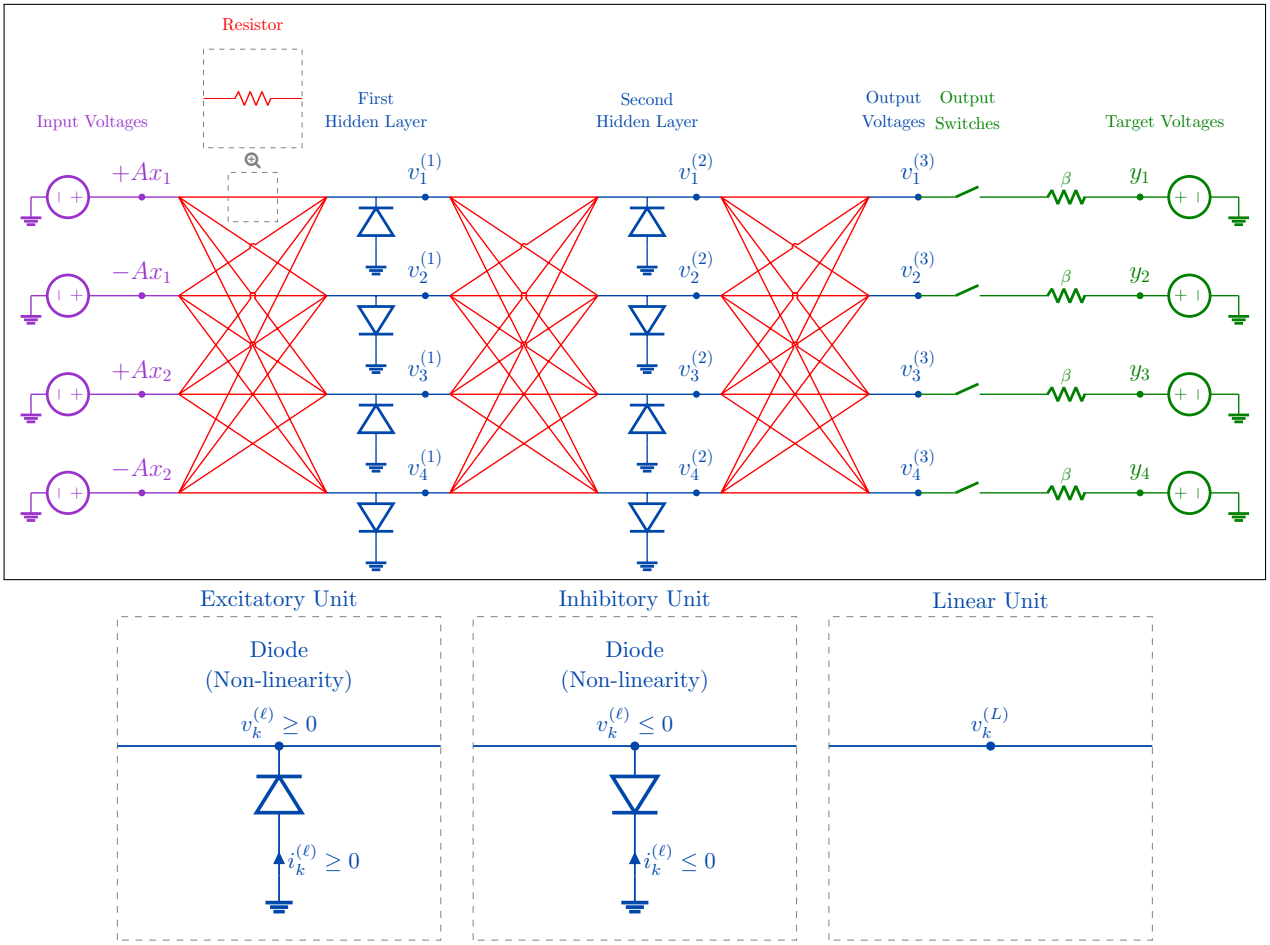

Figure 3. **Top.** A deep resistive network (DRN) with $L = 3$ layers. Input voltage sources are set to input values: $v_1^{(0)} = Ax_1$, $v_2^{(0)} = -Ax_1$, $v_3^{(0)} = Ax_2$ and $v_4^{(0)} = -Ax_2$, where A is the input amplification factor. At inference, output switches are open. Equilibrium propagation learning requires *nudging* the output node voltages ($v_1^{(3)}$, $v_2^{(3)}$ $v_3^{(3)}$ and $v_4^{(3)}$) towards the target voltages (y_1, y_2, y_3 and y_4), which is achieved by closing the output switches. In the DRN architecture, the update rule for a given unit prescribed by exact coordinate descent depends only on the states of the units of the previous layer and the next layer. We can thus update the even layers ($\ell = 2$) simulatneously, and then update all the odd layers ($\ell = 1$ and $\ell = 3$) simultaneously. This is called exact block coordinate descent. **Bottom.** To form a nonlinear unit, we place a diode between the unit's node and ground. Depending on the orientation of the diode, the units come in two flavours: excitatory units and inhibitory units.

are linear, i.e. they do not possess diodes. The feasible set (4) corresponding to this DRN architecture is

$$\mathcal{S} = \{v \in \mathbb{R}^{\sum_{\ell=1}^{L} N_\ell} \mid v_k^{(\ell)} \geq 0 \text{ if k is even}, \quad (13)$$

$$v_k^{(\ell)} \leq 0 \text{ if k is odd}, 1 \leq \ell \leq L-1, 1 \leq k \leq N_\ell\}. \quad (14)$$

Input voltage sources. One constraint with resistive networks in general, and deep resistive networks in particular, is the non-negativity of the weights - a conductance is non-negative. By using both excitatory units and inhibitory units in the hidden layers, we have partially overcome this constraint. To further enhance the representational capacity of the DRN, we also double the number of units in the input layer – i.e. we choose $N_0 = 2\dim(x)$ where $\dim(x)$ is the dimension of input x – and we set the input voltage sources such that $v_{2k}^{(0)} = -v_{2k-1}^{(0)}$ for each $1 \leq k \leq N_0$. Another constraint with DRNs is the decay in amplitude of the layers' voltage values, as the depth of the network increases. We overcome this constraint by amplifying the input voltages by a factor $A \gg 1$, so that $v_{2k-1}^0 = +Ax_k$ and $v_{2k}^0 = -Ax_k$ for every k, where x_k is the k-th input value ($1 \leq k \leq N_0$).

Output switches. Finally, additional circuitry is needed for equilibrium propagation (EP) learning. Each output node is linked to ground via a switch in series with a resistor and a voltage source. The voltage sources are used to set

the desired output values, playing the role of targets. The resistors all share a common conductance value $\beta > 0$. During inference, the switches are open, whereas in the training phase of EP, the switches are closed to drive the state of output units towards the desired output values – see Appendix B for a brief presentation of EP (Scellier & Bengio, 2017).

3.2. A Fast Algorithm to Simulate Deep Resistive Networks

In a DRN, the update rules (Theorem 2) for the units take the following form. For every (ℓ, k) such that $1 \leq \ell \leq L-1$ and $1 \leq k \leq N_\ell$, the update rule for $v_k^{(\ell)}$ is

$$p_k^{(\ell)} := \frac{\sum_{j=1}^{N_{\ell-1}} g_{jk}^{(\ell)} v_j^{(\ell-1)} + \sum_{j=1}^{N_{\ell+1}} g_{kj}^{(\ell+1)} v_j^{(\ell+1)}}{\sum_{j=1}^{N_{\ell-1}} g_{jk}^{(\ell)} + \sum_{j=1}^{N_{\ell+1}} g_{kj}^{(\ell+1)}} \tag{15}$$

$$v_k^{(\ell)} \leftarrow \begin{cases} \max\left(0, p_k^{(\ell)}\right) & \text{if k is even (excitatory unit),} \\ \min\left(0, p_k^{(\ell)}\right) & \text{if k is odd (inhibitory unit).} \end{cases} \tag{16}$$

Assuming that the output switches are closed, the update rule for the output unit $v_k^{(L)}$ is

$$v_k^{(L)} \leftarrow \frac{\sum_{j=1}^{N_{\ell-1}} g_{jk}^{(L)} v_j^{(L-1)} + \beta y_k}{\sum_{j=1}^{N_{\ell-1}} g_{jk}^{(L)} + \beta}. \tag{17}$$

If the output switches are open, Equation (17) still holds by setting $\beta = 0$.

We now derive a specialized, fast algorithm to compute the steady state of a DRN. Since the update rule for $v_k^{(\ell)}$ depends only on the state of the units in layers $\ell - 1$ and $\ell + 1$, and since this is true for all the units in layer ℓ, we may update all these units simultaneously rather than sequentially. Pushing this idea further, we can partition the layers of the network in two groups: the group of layers of even index (even ℓ) and the group of layers of odd index (odd ℓ). Since the update rules for the odd layers depend only on the state of even layers, and vice versa, we can compute the steady state of a DRN by updating alternatively the layers of odd indices (given the state of the layers of even indices fixed) and the layers of even indices (given the state of the layers of odd indices fixed). We obtain an 'exact block coordinate descent' algorithm to simulate DRNs. The property of DRNs that make it possible is the bipartite structure of their graph, where the layers of even indices constitute one set, and the layers of odd indices constitute the other set.

Importantly, equations (15) and (17) can be written in matrix-vector form. Each step of our exact block coordinate descent algorithm (updating half of the layers) thus consists in performing $\sim L$ matrix-vector multiplications, $\sim L/2$ divisions and $\sim L/2$ clipping operations. This makes it an algorithm ideally suited to run on parallel computing platforms such as GPUs.

We note that it is possible to further speed up the simulations of DRNs by computing the row-wise sums of the weight matrices (used at the denominator of the update rules) only once for each inference (steady state computation), instead of computing it at each iteration of the algorithm.

4. Simulations

We use our exact block coordinate descent algorithm to train deep resistive networks (DRNs) with equilibrium propagation (EP) (Scellier & Bengio, 2017) on the MNIST classification task. We train DRNs of one, two and three hidden layers (each comprising 1024 units), denoted DRN-1H, DRN-2H and DRN-3H, respectively. We also train another two DRNs with a single hidden layer each, compring 32784 units and 100 units, denoted DRN-XL and DRN-XS, respectively. The DRN-XS model has the same architecture as in Kendall et al. (2020), while the DRN-XL model is 325x larger. EP is presented in Appendix B and a full description of the DRN models and the hyperparameters used for training are provided in Appendix C.[2]

As a baseline for EP, we also trained DRNs with a version of backpropagation (BP) described in Appendix C. This version of BP is only applicable in simulations and cannot be implemented on analog hardware in any obvious way, as it requires backpropagating through the trajectory of configurations obtained by our exact block coordinate descent algorithm.

Table 1 shows the results.

The largest network that we train (the DRN-XL model) has 327 times as many parameters as the DRN-XS architecture used in Kendall et al. (2020) (51.7M vs 0.16M). We train it for 10 times as many epochs (100 vs 10) and the total duration of training is 16 times shorter (10 hours 27 min vs 1 week). Thus, our network-size-to-epoch-duration ratio is 50000x larger. The performance obtained with the DRN-XL model is also significantly better (1.33% vs 3.43% test error rate).

For all the simulations, we used mini-batches of size 4, because we found that this batch size yields the best results (test error rate). However, the simulation times of the DRN-XS, -1H, -2H and -3H models can be significantly reduced by using larger batch sizes.

[2]The code to reproduce the results is available at https://github.com/rain-neuromorphics/energy-based-learning

Table 1. We trained five deep resistive network (DRN) architectures (XS, XL, 1H, 2H and 3H) on MNIST. The size of the network, as measured per the number of weights (in millions), is written in brackets. Alg refers to the training algorithm: equilibrium propagation (EP) or backpropagation (BP). Our exact block coordinate descent method was used to compute the DRN steady states. Test refers to the test error rates (in %). For each experiment, we performed five runs and we reported the mean values and std values. We also reported the number of epochs of training and the wall-clock time (WCT). A full description of the models with the hyperparameters used for training are reported in Table 2 of Appendix C. We also report the results of the SPICE-based simulations of Kendall et al. (2020) used as a baseline, denoted SPICE XS.

DRN	ALG	TEST (%)	EPOCHS	WCT
SPICE XS (0.16M)	EP	3.43	10	1 WEEK
XS (0.16M)	EP	3.46 ± 0.07	10	0:30
	BP	3.30 ± 0.13	10	0:32
XL (51.7M)	EP	1.33 ± 0.02	100	10:27
	BP	1.30 ± 0.03	100	8:19
1H (1.6M)	EP	1.57 ± 0.07	50	2:36
	BP	1.54 ± 0.04	50	2:48
2H (2.7M)	EP	1.48 ± 0.05	50	4:29
	BP	1.45 ± 0.08	50	4:53
3H (3.7M)	EP	1.66 ± 0.09	50	6:57
	BP	1.50 ± 0.07	50	7:55

5. Discussion

As nonlinear resistive networks have recently attracted interest as physical (analog) self-learning machines, efficiently simulating these networks has become essential to advance research in this direction. Previous works either used general purpose circuit simulators such as SPICE – which are extremely slow as they were not specifically conceived for the simulations of nonlinear resistive networks – or resorted to linear networks, which are easier to simulate but lack a crucial feature of machine learning: nonlinearity. In this work, we have introduced a methodology specifically tailored for the simulations of nonlinear resistive networks. We have shown that the problem of determining the steady state of a nonlinear resistive network can be formulated as a convex quadratic programming (QP) problem, where the objective function is the power dissipated in the resistors, and the feasible set of node electrical potentials reflects the inequality constraints imposed by the diodes. Using this formulation, we have introduced an efficient algorithm to simulate nonlinear resistive networks with arbitrary network topologies, based on 'exact coordinate descent' (Wright, 2015). In the case of layered networks - deep resistive networks (DRNs) - we took advantage of the bipartite structure of the network to derive an exact block coordinate descent algorithm where half of the coordinates (node electrical potentials) are updated at each step. Each step of our algorithm involves solely matrix-vector multiplications, divisions and clipping, making it ideal to run on parallel processors such as GPUs. Compared to the SPICE-based simulations of Kendall et al. (2020), the largest networks that we trained are 327 times larger, and simulating them (on a single A100 GPU) was 160 times faster. Training larger networks for more epochs also helped us achieve significantly better results on the MNIST dataset (1.33% vs 3.43% test error rate).

Although we have primarily focused on layered architectures (DRNs), our algorithm applies to arbitrary network topologies, including unstructured (disordered) networks (Stern et al., 2022; 2024; Wycoff et al., 2022). In such networks, parallelization is also possible, although it is less straightforward to implement. More generally, our characterization of the steady state as the solution of a quadratic programming (QP) problem with linear constraints offers other options to simulate nonlinear resistive networks. Indeed, QP problems can be solved with various algorithms, such as primal-dual interior point methods (IPM) and sequential quadratic programming (SQP) methods. Many optimization libraries and software packages provide specialized QP solvers. Besides, while in our simulations we have used equilibrium propagation (EP) for training, our exact coordinate descent algorithm for nonlinear resistive networks can be used in conjunction with other learning algorithms such as 'frequency propagation' (Anisetti et al., 2024) and 'agnostic EP' (Scellier et al., 2022). It can also be used in conjunction with e.g. the method proposed by Stern et al. (2024) to mitigate power dissipation.

While our methodology to simulate nonlinear resistive networks is fairly general, it is also important to acknowledge the limitations of our approach. Our methodology is anchored in an assumption of ideality of the circuit elements (resistors, diodes, voltage sources and current sources). Real-world diodes, however, deviate significantly from the ideal model studied here: they have a forward voltage drop, reverse leakage current, non-zero resistance when forward-biased, and a breakdown voltage when reverse-biased. Despite these assumptions, which may seem overly simplistic, we believe that our algorithm can be usefully applied to prototype and explore a wide range of network topologies. Our methodology can also be extended to the simulations of electrical networks composed of elements possessing more realistic i-v curves (e.g. piece-wise linear i-v curves), but we leave this for future work to investigate.

Looking forward, our simulation methodology can foster more rapid progress in nonlinear resistive network research. It offers the perspective to perform large scale simulations of deep resistive networks (DRNs), to enable further assessment of the scalability of such physical (analog) self-learning machines on more complex tasks. In this respect,

a related line of works on continuous Hopfield networks holds promise. Nonlinear resistive networks and DRNs are, indeed, closely related to continuous Hopfield networks (Hopfield, 1984) and their layered version, the 'deep Hopfield network' (DHN) (Scellier & Bengio, 2017). In particular, our exact coordinate descent algorithm for resistive networks is similar in spirit to the asynchronous update scheme for Hopfield networks, and our exact block coordinate descent algorithm for DRNs is similar to the asynchronous energy minimization scheme for DHNs (Scellier et al., 2024) and to block-Gibbs sampling in deep Boltzmann machines (Salakhutdinov & Hinton, 2009). Importantly, unlike Hopfield networks and Boltzmann machines whose energy functions are typically non-convex, the energy function (power dissipation) of a nonlinear resistive network is convex. In practice we find that our DRN simulations require much fewer steps to converge than in DHNs. What is especially interesting is that recent works have shown that DHNs yield promising results on image classification tasks such as CIFAR-10 (Laborieux et al., 2021), CIFAR-100 (Scellier et al., 2024) and ImageNet 32x32 (Laborieux & Zenke, 2022). A more detailed analysis of the similarities and differences between resistive networks and continuous Hopfield networks is provided in Appendix E.

Acknowledgements

The author thanks the anonymous reviewers of ICML for their useful comments, as well as Sid Mishra, Jack Kendall, Maxence Ernoult, Mohammed Fouda, Suhas Kumar and Jeremie Laydevant for useful feedback and discussions.

Impact Statement

This work presents an algorithm that could enable large-scale modelling of energy-efficient hardware for machine learning (ML), and therefore facilitate the development of such hardware. Such energy-efficient hardware would significantly reduce the costs associated with inference and training of ML models, and therefore make advanced AI technologies more affordable and accessible to a wider range of populations and industrial sectors. Such hardware would also open the door to train and deploy significantly larger AI models.

References

Altman, L. E., Stern, M., Liu, A. J., and Durian, D. J. Experimental demonstration of coupled learning in elastic networks. *arXiv preprint arXiv:2311.00170*, 2023.

Anisetti, V. R., Kandala, A., Scellier, B., and Schwarz, J. Frequency propagation: Multimechanism learning in nonlinear physical networks. *Neural Computation*, pp. 1–25, 2024.

Dillavou, S., Stern, M., Liu, A. J., and Durian, D. J. Demonstration of decentralized physics-driven learning. *Physical Review Applied*, 18(1):014040, 2022.

Dillavou, S., Beyer, B. D., Stern, M., Miskin, M. Z., Liu, A. J., and Durian, D. J. Machine learning without a processor: Emergent learning in a nonlinear electronic metamaterial. *arXiv preprint arXiv:2311.00537*, 2023.

Ernoult, M., Grollier, J., Querlioz, D., Bengio, Y., and Scellier, B. Updates of equilibrium prop match gradients of backprop through time in an rnn with static input. *Advances in neural information processing systems*, 32, 2019.

Falk, M., Strupp, A., Scellier, B., and Murugan, A. Contrastive learning through non-equilibrium memory. *arXiv preprint arXiv:2312.17723*, 2023.

Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10): 3088–3092, 1984.

Keiter, E. Xyce: An open source spice engine, Mar 2014. URL https://nanohub.org/resources/20605.

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

Kiraz, F. Z., Pham, D.-K. G., and Desgreys, P. Impacts of feedback current value and learning rate on equilibrium propagation performance. In *2022 20th IEEE Interregional NEWCAS Conference (NEWCAS)*, pp. 519–523. IEEE, 2022.

Laborieux, A. and Zenke, F. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. *Advances in Neural Information Processing Systems*, 35:12950–12963, 2022.

Laborieux, A., Ernoult, M., Scellier, B., Bengio, Y., Grollier, J., and Querlioz, D. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

Laydevant, J., Marković, D., and Grollier, J. Training an ising machine with equilibrium propagation. *Nature Communications*, 15(1):3671, 2024.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Marković, D., Mizrahi, A., Querlioz, D., and Grollier, J. Physics for neuromorphic computing. *Nature Reviews Physics*, 2(9):499–510, 2020.

Martin, E., Ernoult, M., Laydevant, J., Li, S., Querlioz, D., Petrisor, T., and Grollier, J. Eqspike: spike-driven equilibrium propagation for neuromorphic implementations. *Iscience*, 24(3), 2021.

Massar, S. and Mognetti, B. M. Equilibrium propagation: the quantum and the thermal cases. *arXiv preprint arXiv:2405.08467*, 2024.

Millar, W. Cxvi. some general theorems for non-linear systems possessing resistance. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 42(333):1150–1160, 1951.

Oh, S., An, J., Cho, S., Yoon, R., and Min, K.-S. Memristor crossbar circuits implementing equilibrium propagation for on-device learning. *Micromachines*, 14(7):1367, 2023.

Onsager, L. Reciprocal relations in irreversible processes. ii. *Physical review*, 38(12):2265, 1931.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Salakhutdinov, R. and Hinton, G. Deep boltzmann machines. In *Artificial intelligence and statistics*, pp. 448–455. PMLR, 2009.

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

Scellier, B. Quantum equilibrium propagation: Gradient-descent training of quantum systems. *arXiv preprint arXiv:2406.00879*, 2024.

Scellier, B. and Bengio, Y. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

Scellier, B. and Mishra, S. A universal approximation theorem for nonlinear resistive networks. *arXiv preprint arXiv:2312.15063*, 2023.

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

Scellier, B., Ernoult, M., Kendall, J., and Kumar, S. Energy-based learning algorithms for analog computing: a comparative study. *Advances in Neural Information Processing Systems*, 36, 2024.

Stern, M., Dillavou, S., Miskin, M. Z., Durian, D. J., and Liu, A. J. Physical learning beyond the quasistatic limit. *Physical Review Research*, 4(2):L022037, 2022.

Stern, M., Dillavou, S., Jayaraman, D., Durian, D. J., and Liu, A. J. Training self-learning circuits for power-efficient solutions. *APL Machine Learning*, 2(1), 2024.

Vogt, H., Hendrix, M., and Nenzi, P. Ngspice (version 31), 2020. URL http://ngspice.sourceforge.net/docs/ngspice-manual.pdf.

Wang, Q., Wanjura, C. C., and Marquardt, F. Training coupled phase oscillators as a neuromorphic platform using equilibrium propagation. *arXiv preprint arXiv:2402.08579*, 2024.

Watfa, M., Garcia-Ortiz, A., and Sassatelli, G. Energy-based analog neural network framework. *Frontiers in Computational Neuroscience*, 17:1114651, 2023.

Williams, E., Bredenberg, C., and Lajoie, G. Flexible phase dynamics for bio-plausible contrastive learning. In *International Conference on Machine Learning*, pp. 37042–37065. PMLR, 2023.

Wright, S. J. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015.

Wycoff, J. F., Dillavou, S., Stern, M., Liu, A. J., and Durian, D. J. Desynchronous learning in a physics-driven learning network. *The Journal of Chemical Physics*, 156(14), 2022.

Xia, Q. and Yang, J. J. Memristive crossbar arrays for brain-inspired computing. *Nature materials*, 18(4):309–323, 2019.

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

Zucchet, N. and Sacramento, J. Beyond backpropagation: bilevel optimization through implicit differentiation and equilibrium propagation. *Neural Computation*, 34(12): 2309–2346, 2022.

A. Proofs of Theorem 1 and Theorem 2

In this appendix, we prove Theorem 1 and Theorem 2.

First we recall our assumptions about the behaviour of individual devices. We consider four types of elements: linear resistors, diodes, voltage sources and current sources, with the following current-voltage (i-v) characteristics:

- A linear resistor follows Ohm's law, i.e. $i = gv$ where g is the conductance of the resistor ($g = 1/r$ where r is the resistance).
- A diode satisfies $i = 0$ for $v \leq 0$, and $v = 0$ for $i > 0$.
- A voltage source satisfies $v = v_0$ for some constant v_0, regardless of i.
- A current source satisfies $i = i_0$ for some constant i_0, regardless of v.

A.1. Proof of Theorem 1

We denote v^{VS} as the set of voltages across voltage sources, and i^{CS} as the set of currents across current sources. For clarity, we restate Theorem 1.

Theorem 1 (Convex QP formulation)**.** *Consider a nonlinear resistive network with N nodes, and denote $v = (v_1, v_2, \ldots, v_N)$ the vector of node electrical potentials*[3]*. Under the assumption of ideality, the steady state configuration of node electrical potentials, denoted $v_\star$, satisfies*

$$v_\star = \operatorname*{arg\,min}_{v \in \mathcal{S}} E(v), \tag{1}$$

where $E : \mathbb{R}^N \to \mathbb{R}$ is defined by

$$E(v_1, \ldots, v_N) := \frac{1}{2} \sum_{(j,k) \in \mathcal{B}_{\mathrm{R}}} g_{jk} \left(v_j - v_k\right)^2 \tag{2}$$

$$+ \sum_{(j,k) \in \mathcal{B}_{\mathrm{CS}}} i^{\mathrm{CS}}_{jk} \left(v_j - v_k\right), \tag{3}$$

and $\mathcal{S}$ is defined as:

$$\mathcal{S} := \{(v_1, v_2, \ldots, v_N) \in \mathbb{R}^N, \tag{4}$$

$$v_j \leq v_k \quad \forall (j,k) \in \mathcal{B}_{\mathrm{D}}, \tag{5}$$

$$v_j = v_k + v^{\mathrm{VS}}_{jk} \quad \forall (j,k) \in \mathcal{B}_{\mathrm{VS}}\}. \tag{6}$$

Proof of Theorem 1. Suppose there exists a configuration of branch voltages and branch currents that satisfies all the current-voltage equations in every branch, as well as Kirchhoff's current law (KCL) and Kirchhoff's voltage law (KVL). We call this configuration the 'steady state' and denote $v_\star = (v_1, \ldots, v_N)$ the corresponding configuration of node electrical potentials, where N is the number of nodes in the network. For every branch $(j,k) \in \mathcal{B}_{\mathrm{R}}$ (resp. $\mathcal{B}_{\mathrm{D}}$, $\mathcal{B}_{\mathrm{VS}}$), we denote i^{R}_{jk} (resp. i^{D}_{jk}, i^{VS}_{jk}) the current through it. The resistor's equation (Ohm's law) imposes that

$$\forall (j,k) \in \mathcal{B}_{\mathrm{R}}, \qquad i^{\mathrm{R}}_{jk} = g_{jk}(v_j - v_k). \tag{18}$$

Let us define the functions G_{jk} (for $(j,k) \in \mathcal{B}_{\mathrm{D}}$) and H_{jk} (for $(j,k) \in \mathcal{B}_{\mathrm{VS}}$) as

$$\forall (j,k) \in \mathcal{B}_{\mathrm{D}}, \quad G_{jk}(v) := v_j - v_k \qquad \text{and} \qquad \forall (j,k) \in \mathcal{B}_{\mathrm{VS}}, \quad H_{jk}(v) := v_j - v_k - v^{\mathrm{VS}}_{jk}. \tag{19}$$

The diode equations impose that

$$\forall (j,k) \in \mathcal{B}_{\mathrm{D}}, \qquad G_{jk}(v) \leq 0, \qquad i^{\mathrm{D}}_{jk} \geq 0, \qquad i^{\mathrm{D}}_{jk} \cdot G_{jk}(v) = 0. \tag{20}$$

[3]The electrical potentials are defined up to a constant, so we may assume, for instance, $v_1 = 0$.

The voltage source equality constraints impose that

$$\forall (j,k) \in \mathcal{B}_{\mathrm{VS}}, \qquad H_{jk}(v) = 0. \tag{21}$$

Next, let k be a node such that $1 \leq k \leq N$, and consider all the branches connected to node k. These branches may include resistors, current sources, diodes and voltage sources. KCL applied to node k reads

$$\sum_{j:(j,k)\in\mathcal{B}_{\mathrm{R}}} i_{jk}^{\mathrm{R}} + \sum_{j:(j,k)\in\mathcal{B}_{\mathrm{CS}}} i_{jk}^{\mathrm{CS}} + \sum_{j:(j,k)\in\mathcal{B}_{\mathrm{D}}} i_{jk}^{\mathrm{D}} + \sum_{j:(j,k)\in\mathcal{B}_{\mathrm{VS}}} i_{jk}^{\mathrm{VS}} = 0. \tag{22}$$

Let us introduce the following function L, which is a function of the node electrical potentials (v), the currents through the diodes (i^{D}), and the currents through the voltage sources (i^{VS}),

$$L(v, i^{\mathrm{D}}, i^{\mathrm{VS}}) := E(v) + \sum_{(j,k)\in\mathcal{B}_{\mathrm{D}}} i_{jk}^{\mathrm{D}} G_{jk}(v) + \sum_{(j,k)\in\mathcal{B}_{\mathrm{VS}}} i_{jk}^{\mathrm{VS}} H_{jk}(v). \tag{23}$$

Using the expressions of the energy function (2) and the functions G_{jk} and H_{jk} from (19), KCL (22) can be rewritten in terms of L as

$$\nabla_v L(v, i_{jk}^{\mathrm{D}}, i_{jk}^{\mathrm{VS}}) = 0. \tag{24}$$

Equations (20), (21) and (24) fully characterize the steady state of the nonlinear resistive network. These equations constitute the set of Karush-Kuhn-Tucker (KKT) conditions associated with the constrained optimization problem of Eq. (1). L is the Lagrangian, and the currents i^{D} and i^{VS} are the Lagrange multipliers associated with the inequality and equality constraints, respectively. Equations (20) and (21) constitute the primal feasibility condition (ensuring v belongs to the feasible set $\mathcal{S}$), the dual feasibility condition (ensuring the Lagrange multipliers associated with the inequality constraints are non-negative), and the complementary slackness for inequality constraints (either the constraint is 'binding' or the associated Lagrange multiplier is zero). Finally, (24) is the stationarity condition of the Lagrangian.

From the above analysis, it follows that the steady state of the network (if it exists) satisfies the KKT conditions associated with the constrained optimization problem (1). Conversely, a configuration of node voltages and branch currents (diode branches and voltage source branches) that satisfies the KKT conditions is a steady state.

To conclude, we note that, as a sum of convex functions, the energy function E is convex. Therefore the KKT conditions are equivalent to global optimality. This implies that a steady state exists if and only if the feasible set is non-empty, in which case the corresponding configuration of node voltages is a global minimum of the convex optimization problem. □

A.2. Proof of Theorem 2

The optimization problem is a quadratic programming (QP) problem with linear constraints. We solve it using exact coordinate descent.

Theorem 2 (Exact coordinate descent)**.** *Let $v = (v_1, \ldots, v_N) \in \mathcal{S}$. Let k be an internal node of the network, i.e. a node that does not belong to the connected component of voltage sources. Define*

$$p_k := \frac{\sum_{j\in\mathcal{B}_{\mathrm{R}}} g_{kj} v_j + \sum_{j\in\mathcal{B}_{\mathrm{CS}}} i_{jk}^{\mathrm{CS}}}{\sum_{j\in\mathcal{B}_{\mathrm{R}}} g_{kj}} \tag{7}$$

and

$$v_k^{\min} := \max_{j:(j,k)\in\mathcal{B}_{\mathrm{D}}} v_j, \qquad v_k^{\max} := \min_{j:(k,j)\in\mathcal{B}_{\mathrm{D}}} v_j, \tag{8}$$

$$v_k' := \min\left(\max\left(v_k^{\min}, p_k\right), v_k^{\max}\right), \tag{9}$$

$$v' := (v_1, \ldots, v_{k-1}, v_k', v_{k+1}, \ldots, v_N). \tag{10}$$

Then, among all configurations $v'' \in \mathcal{S}$ of the form $v'' = (v_1, \ldots, v_{k-1}, v_k'', v_{k+1}, \ldots, v_N)$, the configuration v' is the one with the lowest energy, i.e.

$$v_k' = \operatorname*{arg\,min}_{v_k''\in\mathcal{S}} E(v_1, \ldots, v_{k-1}, v_k'', v_{k+1}, \ldots, v_N). \tag{11}$$

In particular $v' \in \mathcal{S}$ and $E(v') \leq E(v)$.

Proof of Theorem 2. The energy function, as a function of v_k, is a second-order polynomial of the form:

$$E(v_k) = \frac{1}{2} \sum_{j:(j,k)\in\mathcal{B}_{\mathrm{R}}} g_{jk} \left(v_j - v_k\right)^2 + \sum_{j:(j,k)\in\mathcal{B}_{\mathrm{CS}}} i_{jk}^{\mathrm{CS}} (v_j - v_k) \tag{25}$$

$$= \underbrace{\left(\frac{1}{2} \sum_{j:(j,k)\in\mathcal{B}_{\mathrm{R}}} g_{jk} \right)}_{=:a_k} v_k^2 - \underbrace{\left(\sum_{j:(j,k)\in\mathcal{B}_{\mathrm{R}}} g_{kj} v_j + \sum_{j:(j,k)\in\mathcal{B}_{\mathrm{CS}}} i_{jk}^{\mathrm{CS}} \right)}_{=:b_k} v_k + \text{constant} \tag{26}$$

The minimum (in $\mathbb{R}$) of this second-order polynomial is achieved at the point $v_k = p_k$, defined as

$$p_k := -\frac{b_k}{2a_k} = \frac{\sum_{j:(j,k)\in\mathcal{B}_{\mathrm{R}}} g_{kj} v_j + \sum_{j:(j,k)\in\mathcal{B}_{\mathrm{CS}}} i_{jk}^{\mathrm{CS}}}{\sum_{j:(j,k)\in\mathcal{B}_{\mathrm{R}}} g_{jk}}. \tag{27}$$

However, the range of feasible values for v_k (for which the corresponding configurations are in the feasible set $\mathcal{S}$) is not $\mathbb{R}$, but $(v_k^{\min}, v_k^{\max})$, where

$$v_k^{\min} := \max_{j:(j,k)\in\mathcal{B}_{\mathrm{D}}} v_j, \qquad v_k^{\max} := \min_{j:(k,j)\in\mathcal{B}_{\mathrm{D}}} v_j. \tag{28}$$

It is easily checked that the value $v_k' \in (v_k^{\min}, v_k^{\max})$ that achieves the minimum of the above second-order polynomial (given other variables fixed) is

$$v_k' := \min\left(\max\left(v_k^{\min}, p_k\right), v_k^{\max}\right). \tag{29}$$

□

B. Equilibrium Propagation

In this appendix, we briefly present the equilibrium propagation (EP) algorithm used in section 4 to train the deep resistive network (DRN). EP is a method for extracting the weight gradients of arbitrary cost functions in arbitrary energy-based systems, that is, systems that possess an energy function and seek an energy minimum (Scellier & Bengio, 2017) (see also Scellier (2021) for a physics-oriented presentation of EP). Besides the nonlinear resistive networks considered in this work, EP has been used in continuous Hopfield networks (see Appendix E), Ising machines (Laydevant et al., 2024), coupled phase oscillators (Wang et al., 2024) and spiking networks (Martin et al., 2021). A variant of EP has also been used in elastic networks (Altman et al., 2023). Most recently, EP has been extended to quantum systems (Massar & Mognetti, 2024; Scellier, 2024).

B.1. EP in Nonlinear Resistive Networks

Consider a nonlinear resistive network with arbitrary topology, as described in Section 2 (see Figure 4, top). In this electrical network, a set of voltage sources serves as 'input variables' with voltages set to the input values, while other branch voltages and currents settle to a steady state, i.e. a minimum of the energy function (2), characterized by

$$v_\star := \underset{v \in \mathcal{S}}{\arg\min}\, E(v). \tag{30}$$

A subset of these branches functions as 'output variables', with their voltages used as the network's prediction.[4] We view this input-output function implemented by the network as being parameterized by the conductances of the resistors, while the diodes introduce nonlinearities. The goal is to adjust the conductance values of the resistors (the 'trainable weights') so the network implements a desired input-output function. Several learning algorithms for such resistive networks have been proposed (Kendall et al., 2020; Dillavou et al., 2022; Anisetti et al., 2024) based on EP. In this work, we used EP for its simplicity and superior performance compared to other contrastive learning methods (Scellier et al., 2024).

EP involves augmenting the network by adding a branch between each pair of output nodes. Each branch consists of a voltage source, a resistor and a switch in series (Figure 4, bottom left). All output resistors share a common conductance value, $\beta > 0$. The switches have two states: they are either all open (the 'free state') or all closed (the 'nudge state'). In the free state, no current flows through these output branches, leaving the network's state unchanged. In the nudge state, however, currents flow through these output branches, influencing the network's state. Mathematically, in the nudge state, the network's energy function E (defined in Theorem 1) is augmented by a term equal to the power dissipated in the output resistors, $\frac{1}{2}\sum_{(j,k)\in\mathcal{B}_O} \beta\,(v_{jk} - y_{jk})^2$, with $\mathcal{B}_O$ denoting the set of output branches, and y_{jk} the voltage across the voltage source of output branch (j,k). Thus, in the nudge state, the total energy function (total power dissipation) of the augmented network is

$$F(\beta, v) := E(v) + \beta C(v), \tag{31}$$

where

$$C(v) := \frac{1}{2} \sum_{(j,k)\in\mathcal{B}_O} (v_{jk} - y_{jk})^2 . \tag{32}$$

Importantly, the function C appearing in the 'total energy function' is also the cost function that the network aims to minimize from a learning perspective, representing the squared error between the model prediction and the desired output. In the free state (when the output switches are all open), the energy function of the network is $F(0, v) = E(v)$.

EP operates as follows:

1. Pick an input-output pair (x, y) from the training dataset.

2. Set input voltage sources to the input values (x) and open all the output switches, allowing the network to reach the free state:

$$v_\star^0 := \underset{v \in \mathcal{S}}{\arg\min}\, F(0, v) = v_\star. \tag{33}$$

[4] For example, in the DRN architecture of Figure 3, input voltage sources link input nodes to ground, and output branches link output nodes to ground.

3. Set output voltage sources to the desired output values (y) and close all the output switches, allowing the network to reach the nudge state:

$$v_\star^\beta := \arg\min_{v \in \mathcal{S}} F(\beta, v). \tag{34}$$

4. For every branch (j, k) containing a variable resistor, update the conductance g_{jk} according to the contrastive rule:

$$\Delta g_{jk} = \frac{\eta}{\beta}\left[\frac{\partial F}{\partial g_{jk}}(0, v_\star^0) - \frac{\partial F}{\partial g_{jk}}(\beta, v_\star^\beta)\right] \tag{35}$$

$$= \frac{\eta}{2\beta}\left[\left(v_{jk}^0\right)^2 - \left(v_{jk}^\beta\right)^2\right], \tag{36}$$

where $\eta > 0$ is the learning rate, v_{jk}^0 denotes the voltage across branch (i, j) in the free state, and v_{jk}^β denotes the voltage in the nudge state. We have used the explicit form of the energy function, as in Eq. (2).

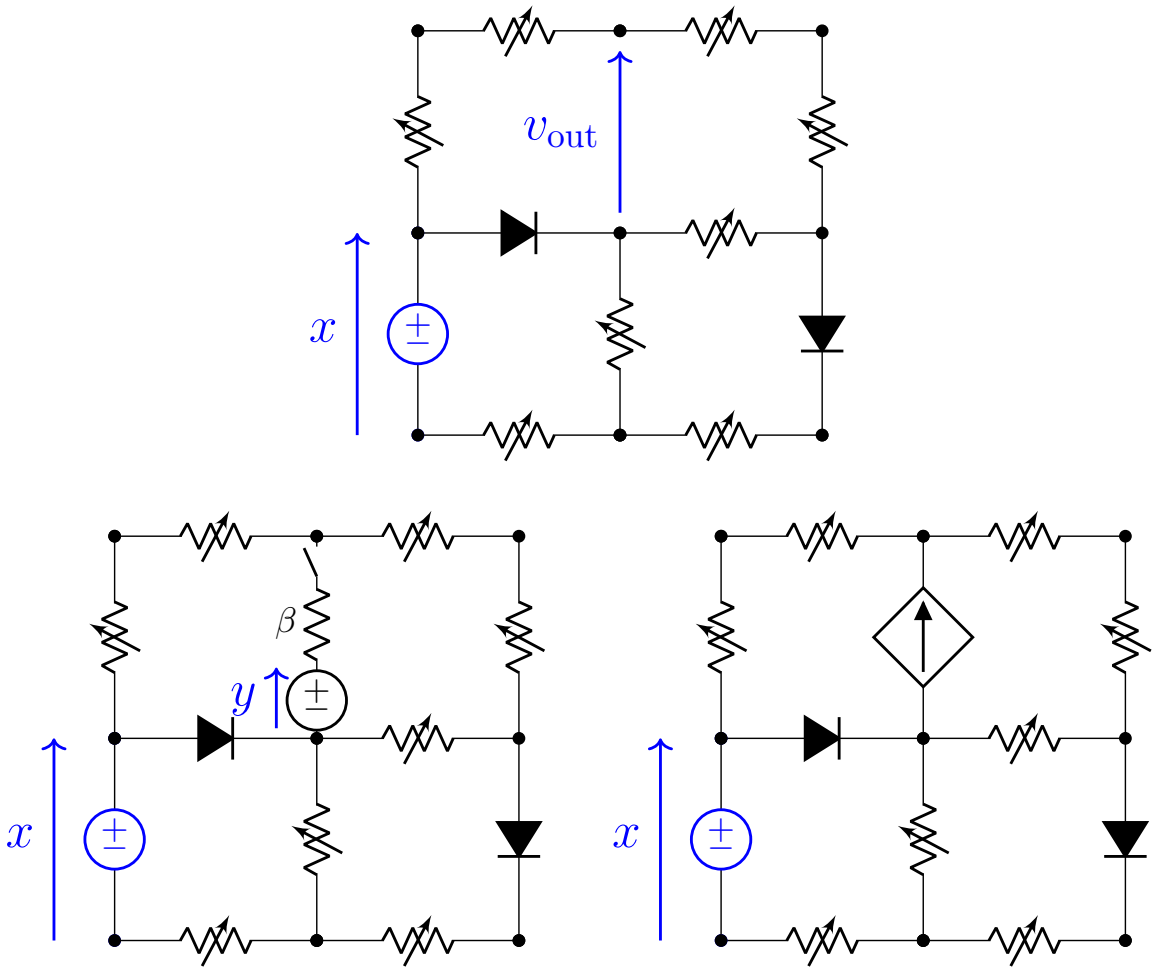

Figure 4. Training a nonlinear resistive network with equilibrium propagation (EP). **Top**. Input voltage x is supplied to the network, and the output voltage v_{out} is measured. Diodes implement nonlinearities and variable resistors implement the 'trainable weights'. **Bottom**. Two methods to implement 'nudging' for EP learning. In the first method (**left**), a voltage source set to desired output y, a resistor of conductance β and a switch are in series in the output branch. Closing the switch injects a current proportional to the prediction error, $i_{\text{out}} = \beta(y - v_{\text{out}})$. One caveat of this method is that the nudging parameter β (the conductance) is necessarily positive. In the second method (**right**), a current source injects a current $i_{\text{out}} = \beta(y - v_{\text{out}})$ in the output branch, allowing for the use of a negative β.

B.2. Equilibrium Propagation Formulas

The central result of EP is that the learning rule (35) approximates one step of gradient descent on the cost function C (Scellier & Bengio, 2017). Specifically,

$$\Delta g_{jk} = -\eta \frac{\partial C(v_\star^0)}{\partial g_{jk}} + O(\beta) \tag{37}$$

when $\beta \to 0$. More specifically, the learning rule (35) performs one step of gradient descent on a surrogate function $\mathcal{L}_\beta$ that approximates the true cost function C (Scellier et al., 2024), i.e.

$$\Delta g_{jk} = -\eta \frac{\partial \mathcal{L}_\beta}{\partial g_{jk}} \qquad \text{where} \qquad \mathcal{L}_\beta = C(v_\star^0) + O(\beta) \quad \text{when} \quad \beta \to 0. \tag{38}$$

The surrogate function $\mathcal{L}_\beta$ is the contrastive function defined as

$$\mathcal{L}_\beta := \frac{G(\beta) - G(0)}{\beta}, \qquad \text{where} \qquad G(\beta) := F(\beta, v_\star^\beta) = \min_{v \in \mathcal{S}} F(\beta, v). \tag{39}$$

EP has been found to work better in practice when employing a negative β in the nudge state, rather than a positive one (Scellier et al., 2024). This can be explained as the contrastive function of EP is an upper bound of the true cost function when β is negative, and a lower bound when it is positive, i.e.

$$\mathcal{L}_\beta \leq C(v_\star^0) \leq \mathcal{L}_{-\beta}, \qquad \forall \beta > 0. \tag{40}$$

A centered version of EP, introduced earlier in Laborieux et al. (2021), combines a negatively-perturbed state $v_\star^{-\beta}$ with a positively-perturbed one $v_\star^{+\beta}$, proving more effective. This can be explained as the contrastive function $\mathcal{L}_\beta^{\text{centered}}$ of this centered variant approximates the true cost function to second order in β, i.e.

$$\mathcal{L}_\beta^{\text{centered}} := \frac{G(\beta) - G(-\beta)}{2\beta} = C(v_\star^0) + O(\beta^2). \tag{41}$$

This is the version of EP that we used in our simulations (Section 4).

Importantly, all these results also hold in the non-ideal regime where the network elements (diodes, voltage sources and current sources) have arbitrary i-v characteristics (not necessarily the i-v characteristics of Figure 1). The only assumption, required to derive Eq. (36), is that the resistors must follow Ohm's law (i.e. have the linear i-v characteristic depicted in Figure 1). We refer to Kendall et al. (2020) for a proof.

The above formulas were proved in Scellier et al. (2024) in the case where the state space is $\mathcal{S} = \mathbb{R}^{\mathbb{N}}$. In Appendix F, we prove that these formulas still hold when the feasible set is defined by equality and inequality constraints.

B.3. Implementation Issues in Hardware

We conclude this Appendix by discussing some challenges related to hardware implementations.

One issue is that the implementation of the nudging proposed in Section B.1 (Figure 4, bottom left) – using a voltage source, a resistor and a switch in series in every output branch – does not allow for negative nudging ($\beta < 0$) since a conductance is always non-negative. To achieve negative nudging, a possibility is to use current sources whose current values are set to $i_{jk} = -\beta(v_{jk} - y_{jk})$, where $\beta \in \mathbb{R}$ (Figure 4, bottom right). This approach was proposed in Kendall et al. (2020). In this method, the free state (inference) is obtained by setting the current sources to zero.

Another implementation issue concerns the contrastive learning rule (35), which requires accessing two different states of the system (free and nudge) to extract the weight gradients. In the experimental demonstration by Dillavou et al. (2022; 2023), two copies of the network are coupled to allow access to both network states (free and nudge) simultaneously, enabling full analog training of the system. A caveat is that this coupled learning approach is subject to mismatches between the two copies of the network, which could cause deviations between measured weight gradients and true weight gradients, potentially affecting optimization. Other practical implementations of EP have resorted to using external memory to store the two states before performing the weight updates (Yi et al., 2023; Laydevant et al., 2024), at the cost of increased energy consumption. Other solutions have been proposed to perform the weight updates using a single network, potentially without

external memory. Williams et al. (2023) propose using only one of the two states, chosen at random at each step, to extract an unbiased (but high variance) estimator of the weight gradients. Anisetti et al. (2024) show that the weight gradients can be extracted using the mean and amplitude of the network response to a sinusoidal nudging signal. Falk et al. (2023) show that the weight update can be performed using a non-equilibrium memory (time convolution of trajectory) of the network state, using integral feedback. Finally, a non-contrastive approach called 'Agnostic EP' (AEP) was introduced in Scellier et al. (2022), where the parameter updates are performed through physical dynamics without relying on external measurement and feedback.

C. Simulation Details

We provide the implementation details to reproduce the results of our simulations of DRNs trained by equilibrium propagation (EP) and backpropagation (BP). The code to reproduce the results is available at https://github.com/rain-neuromorphics/energy-based-learning

MNIST dataset. The MNIST dataset of handwritten digits (LeCun et al., 1998) consists of 60,000 training examples and 10,000 test examples. Each example x in the dataset is a 28×28 grayscale image, accomonpanied by a label $y \in \{0, 1, \dots, 9\}$ indicating the digit represented by the image.

Network architectures. We train five deep resistive network (DRN) models termed DRN-1H, DRN-2H, DRN-3H, DRN-XS and DRN-XL. Each DRN has 1568 input units ($2 \times 28 \times 28$) and 10 output units corresponding to the ten classes of the MNIST dataset. The DRN-kH model has k hidden layers of 1024 units each ($k \in \{1, 2, 3\}$). The DRN-XS model has one hidden layer of 100 units. The DRN-XL model has one hidden layer of 32768 units. Table 2 provides the architectural details of the five DRN models, and the hyperparameters used to obtain the results presented in Table 1.

Initialization of the conductances ('weights'). For two consecutive layers of sizes N_ℓ and $N_{\ell+1}$, the matrix of conductances between these layers is initialized using a modified 'Kaiming uniform' initialization scheme:

$$g_{ij}^{(\ell+1)} = \max(0, w), \qquad w \sim \mathcal{U}(-c, +c), \qquad c = \sqrt{\frac{1}{N_\ell}}. \tag{42}$$

Input amplification factor. We recall that, in a DRN, not only is the number of input nodes (input voltage sources) doubled, but input signals are amplified by a fixed gain factor $A > 0$. For each DRN model (DRN-1H, DRN-2H, DRN-3H, DRN-XS and DRN-XL), the choice of A is reported in Table 2. We note that the values of A are relatively large, e.g. $A = 4000$ for the DRN-3H model. Instead of amplifying input signals of a DRN by a large amplification factor $A \gg 1$, another option is to use bidirectional amplifiers at every layer – see Appendix D for details.

Energy minimization. To compute the network's steady state, we use our exact block coordinate descent algorithm for DRNs (Section 3.2). At every iteration, we first update the layers of even indices (one half of the layers), then we update the layers of odd indices (the other half). We repeat as many iterations as is necessary until convergence to the steady state. In Table 2, T denotes the number of iterations performed during inference (free phase), and K denotes the number of iterations performed during the second (training) phase.

EP training procedure. We trained our networks with equilibrium propagation (EP) and backpropagation (BP). First, we describe EP, as detailed in Appendix B. At each training step, we proceed as follows. First we pick a mini-batch of samples from the training set, x, and their corresponding labels, y. Then we perform T iterations of the block coordinate descent algorithm without nudging ($\beta = 0$). We compute the training loss and training error rate for the current mini-batch, to monitor training. We also store the steady state (free state) $v_\star$. Next, we set the nudging parameter to $\beta > 0$ and we perform a new block-coordinate descent of K iterations, to compute the positively-perturbed state $v_\star^\beta$. Then, we reset the network state to the free state $v_\star$, we set the nudging parameter to $-\beta$, and perform a new block-coordinate descent of K iterations to compute the negatively-perturbed state $v_\star^{-\beta}$. Finally, we update all the conductances in proportion to the 'centered EP' learning rule.

BP training procedure. Similar to EP, we first perform T iterations of block coordinate descent to compute the free state. Then we perform another K steps of block coordinate descent (still using $\beta = 0$) and backpropagate through these K steps to compute the weight gradients. The procedure is equivalent to performing 'truncated backprop through time' where we perform $T + K$ steps in the forward pass, and backpropagate through only the last K steps in the backward pass. This baseline was also used e.g. in Ernoult et al. (2019) and Scellier et al. (2024).

Optimizer and scheduler. We use standard mini-batch gradient descent (SGD) with a mini-batch size of 4. No momentum or weight decay is used. We use a scheduler with a learning rate decay of 0.99 at each training epoch.

Computational resources. The code for the simulations uses PyTorch 1.13.1 and TorchVision 0.14.1. (Paszke et al., 2017). The simulations were carried out on a single Nvidia A100 GPU.

Table 2. Hyper-parameters used for initializing and training the five DRN models (DRN-XS, DRN-XL, DRN-1H, DRN-2H and DRN-3H) to reproduce the results in Table 1. LR stands for 'learning rate'.

	DRN-XS	DRN-XL	DRN-1H	DRN-2H	DRN-3H
INPUT AMPLIFICATION FACTOR (A)	100	800	480	2000	4000
NUDGING (β)	1.0	1.0	1.0	1.0	2.0
NUM. ITERATIONS AT INFERENCE (T)	4	4	4	5	6
NUM. ITERATIONS DURING TRAINING (K)	4	4	4	5	6
LAYER 0 (INPUT) SIZE (N_0)	2-28-28	2-28-28	2-28-28	2-28-28	2-28-28
LAYER 1 SIZE (N_1)	100	32768	1024	1024	1024
LAYER 2 SIZE (N_2)	10	10	10	1024	1024
LAYER 3 SIZE (N_3)				10	1024
LAYER 4 SIZE (N_4)					10
LR WEIGHT 1 & BIAS 1 (η_1)	0.006	0.006	0.006	0.002	0.005
LR WEIGHT 2 & BIAS 2 (η_2)	0.006	0.006	0.006	0.006	0.02
LR WEIGHT 3 & BIAS 3 (η_3)				0.018	0.08
LR WEIGHT 4 & BIAS 4 (η_4)					0.005
LR DECAY	0.99	0.99	0.99	0.99	0.99
MINI-BATCH SIZE	4	4	4	4	4
NUMBER OF EPOCHS	10	100	50	50	50

D. Bidirectional Amplifiers

One challenge with deep resistive networks (DRNs) is the decay in signal amplitude across the network's layers as depth increases. To compensate for this decay, in our construction, we amplify the input voltages by a large gain factor $A \gg 1$ to compensate for this decay. A solution to this problem, proposed in Kendall et al. (2020), is to equip each 'unit' with a 'bidirectional amplifier'. In this appendix, we review this method.

A bidirectional amplifier is a three-terminal device, with bottom (B), left (L) and right (R) terminals. The bottom terminal is linked to ground. The current and voltage states of the left and right terminals, (v_L, i_L) and (v_R, i_R), satisfy the relationship $v_R = a\, v_L$ and $i_R = a\, i_L$, where a is the gain of the bidirectional amplifier. See Figure 5. The bidirectional amplifier amplifies the voltages in the forward direction by a factor $a > 1$ and amplifies the currents in the backward direction by a factor $1/a$. In practice, a bidirectional amplifier can be created by combining a voltage-controlled voltage source (VCVS) and a current-controlled current source (CCCS), as described in Kendall et al. (2020).

Next, we equip each unit with a bidirectional amplifier, defining the unit's state as the voltage after amplification (see Figure 5). We assume that all units in a given layer ℓ use the same gain $a^{(\ell)}$. In this context, Theorem 1 can be restated as follows. For each layer ℓ we define

$$c^{(\ell)} := 1 \times a^{(1)} \times a^{(2)} \times \ldots \times a^{(\ell)}. \tag{43}$$

The energy function E from Eq. (12) becomes

$$E(v) = \frac{1}{2} \sum_{\ell=1}^{L} \sum_{j=1}^{N_{\ell-1}} \sum_{k=1}^{N_\ell} g_{jk}^{(\ell)} \left(\frac{v_j^{(\ell-1)}}{c^{(\ell-1)}} - \frac{v_k^{(\ell)}}{c^{(\ell)}} \right)^2, \tag{44}$$

while the feasible set $\mathcal{S}$ from Eq. (13) remains unchanged:

$$\mathcal{S} = \{ v \in \mathbb{R}^{\sum_{\ell=1}^{L} N_\ell} \mid v_k^{(\ell)} \geq 0 \text{ if k is even}, v_k^{(\ell)} \leq 0 \text{ if k is odd}, 1 \leq \ell \leq L-1, 1 \leq k \leq N_\ell \}. \tag{45}$$

The update rule for $v_k^{(\ell)}$ from Eq. (15) becomes:

$$v_k^{(\ell)} \leftarrow \begin{cases} \max\left(0, p_k^{(\ell)}\right) & \text{if k is even,} \\ \min\left(0, p_k^{(\ell)}\right) & \text{if k is odd,} \end{cases} \qquad \text{where} \qquad p_k^{(\ell)} := \frac{\sum_{j=1}^{N_{\ell-1}} g_{jk}^{(\ell)} a^{(\ell)} v_j^{(\ell-1)} + \sum_{j=1}^{N_{\ell+1}} g_{kj}^{(\ell+1)} \frac{v_j^{(\ell+1)}}{a^{(\ell+1)}}}{\sum_{j=1}^{N_{\ell-1}} g_{jk}^{(\ell)} + \sum_{j=1}^{N_{\ell+1}} g_{kj}^{(\ell+1)}}. \tag{46}$$

The update rule for the output layer from Eq. (17) becomes:

$$v_k^{(L)} \leftarrow \frac{\sum_{j=1}^{N_{\ell-1}} g_{jk}^{(L)} a^{(L)} v_j^{(L-1)} + \beta y_k}{\sum_{j=1}^{N_{\ell-1}} g_{jk}^{(L)} + \beta}. \tag{47}$$

The learning rule for the conductances of a DRN with bidirectional amplifiers also needs slight modification. For the conductance between unit j of layer $\ell - 1$ and unit k of layer ℓ, we denote:

$$g_{jk} := g_{jk}^{(\ell)}, \qquad \widetilde{v}_j := \frac{v_j^{(\ell-1)}}{c^{(\ell-1)}}, \qquad \widetilde{v}_k := \frac{v_k^{(\ell)}}{c^{(\ell)}}. \tag{48}$$

Then the learning rule reads:

$$\Delta g_{jk} = \frac{\eta}{2\beta} \left[\left(\widetilde{v}_j^0 - \widetilde{v}_k^0 \right)^2 - \left(\widetilde{v}_j^\beta - \widetilde{v}_k^\beta \right)^2 \right], \tag{49}$$

where $\widetilde{v}^0$ and $\widetilde{v}^\beta$ denote the free state and nudged state values, respectively. If bidirectional amplifiers are removed from the network (choosing $a^{(\ell)} = 1$ for all ℓ), then $c^{(\ell)} = 1$ for all ℓ, and the original formulas from Section 3 and Appendix B are recovered.

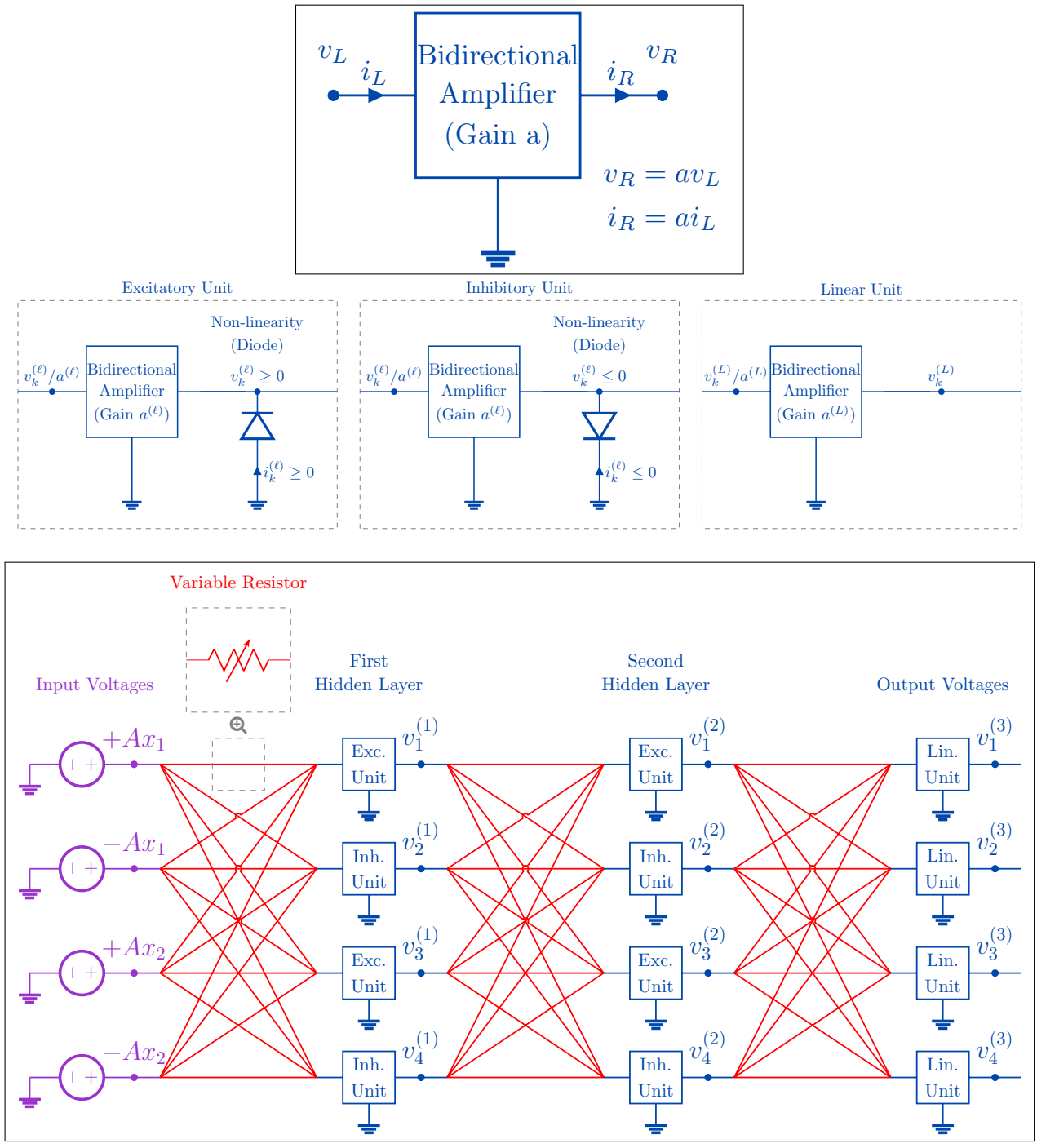

Figure 5. **Top.** Bidirectional amplifier with gain a. The right terminal voltage (v_R) is related to the left terminal voltage (v_L) by $v_R = av_L$, where a is a gain factor. The left terminal current (i_L) is related to the right terminal current (i_R) by $i_L = \frac{1}{a} i_R$. **Middle.** A unit consisting of a bidirectional amplifier, possibly followed by a diode between the unit's node and ground. Units come in two types: excitatory and inhibitory, depending on the diode's orientation. **Bottom.** A deep resistive network with bidirectional amplifiers.

E. Continuous Hopfield Networks

In this appendix, we discuss the similarities and differences between resistive networks and continuous Hopfield networks (CHNs). First, we present the CHN model and its layered version, the deep Hopfield network (DHN). Then, we show that nonlinear resistive networks can be cast as CHNs. Finally, we compare DRNs and DHNs through simulations.

E.1. Continuous Hopfield Networks and Deep Hopfield Networks

The continuous Hopfield network (CHN), introduced by Hopfield (1984), is inspired by the Ising model of spin glasses. A CHN consists of continuous-valued units (s_k) interconnected by bidirectional weights (w_{jk}). Each unit k has an associated bias (b_k), and a nonlinear activation function $f_k : \mathbb{R} \to \mathbb{R}$, assumed to be increasing. The network's state is represented by the vector $s = (s_1, s_2, \dots, s_N)$, where N is the number of units. The energy function of the network, called the 'Hopfield energy function', is defined as

$$E_{\text{Hopfield}}(s_1, \dots, s_N) := \sum_k \underbrace{\int_0^{s_k} f_k^{-1}(u)du}_{:=\Phi_k(s_k)} - \sum_{j<k} w_{jk} s_j s_k - \sum_k b_k s_k. \tag{50}$$

Since the nonlinear activation function f_k is increasing, its inverse function f_k^{-1} is also increasing, making Φ_k convex. This implies that the function $s_k \mapsto E_{\text{Hopfield}}(s_1, \dots, s_N)$ is convex, given the state of other units fixed. The value of s_k that minimizes this function satisfies

$$\partial_{s_k} E_{\text{Hopfield}}(s_1, \dots, s_N) = 0 \quad \iff \quad f_k^{-1}(s_k) - \sum_{j \neq k} w_{jk} s_j - b_k = 0 \tag{51}$$

$$\iff \quad s_k = f_k\left(\sum_{j \neq k} w_{jk} s_j + b_k\right). \tag{52}$$

Thus, the Hopfield energy function can be minimized via exact coordinate descent, where each step updates one unit k according to (52) – similar to our algorithm for nonlinear resistive network (Theorem 2).

In the CHN model of Scellier & Bengio (2017) where equilibrium propagation was first illustrated, the nonlinear activation function is the 'hard sigmoid'

$$f_k(s_k) := \min(\max(0, s_k), 1), \qquad 1 \leq k \leq N. \tag{53}$$

The term $\Phi_k(s_k)$ in the Hopfield energy function (50) becomes

$$\Phi_k(s_k) := \int_0^{s_k} f_k^{-1}(u)du = \begin{cases} +\infty & \text{if} \quad s_k \leq 0, \\ s_k^2 & \text{if} \quad 0 \leq s_k \leq 1, \\ +\infty & \text{if} \quad 1 \leq s_k. \end{cases} \tag{54}$$

In this context, the steady state of the CHN can be cast as the solution of the following constrained minimization problem:

$$s_\star = \underset{s \in \mathcal{S}_{\text{Hopfield}}}{\arg\min} \; E^{\text{ideal}}_{\text{Hopfield}}(s), \tag{55}$$

where $E^{\text{ideal}}_{\text{Hopfield}}$ is the Hopfield energy function corresponding to the hard sigmoid activation function (53), and $\mathcal{S}^{\text{ideal}}_{\text{Hopfield}}$ is the set of feasible configurations:

$$E^{\text{ideal}}_{\text{Hopfield}}(s_1, \dots, s_N) := \frac{1}{2}\sum_k s_k^2 - \sum_{(j,k)} w_{jk} s_j s_k - \sum_k b_k s_k, \tag{56}$$

$$\mathcal{S}^{\text{ideal}}_{\text{Hopfield}} := \{s = (s_1, s_2, \dots, s_N) \in \mathbb{R}^N \mid 0 \leq s_k \leq 1, \quad 1 \leq k \leq N\}. \tag{57}$$

Although the Hopfield energy function is convex in s_k (given the other units fixed), it is not necessarily convex in the overall state $s = (s_1, s_2, \dots s_N)$. This non-convexity leads to the multi-stability issue (discussed by Wang et al. (2024) in the context of coupled phase oscillators). In contrast, nonlinear resistive networks have a convex energy function, resulting in a unique steady state (Theorem 1).

The units of a CHN can be organised into layers to form a 'deep Hopfield network' (DHN). Similar to the deep resistive network (DRN), the bipartite structure of a DHN allows for a block coordinate descent method, where half of the units are updated in parallel at each step – either the layers of odd index or the layers of even index. This method, used in the simulations of Scellier et al. (2024), is akin to the block Gibbs sampling procedure for deep Boltzmann machines (Salakhutdinov & Hinton, 2009).

E.2. Casting Resistive Networks as Continuous Hopfiled Networks

Next we show that a nonlinear resistive network can be cast as a CHN. First we recall the energy function of a nonlinear resistive network (Theorem 1):

$$E_{\text{resistive}} := \frac{1}{2} \sum_{(j,k) \in \mathcal{B}_{\text{R}}} g_{jk} \left(v_j - v_k\right)^2 + \sum_{(j,k) \in \mathcal{B}_{\text{CS}}} i_{jk}^{\text{CS}} \left(v_j - v_k\right). \tag{58}$$

We define G_j as the sum of conductances linked to node j, and I_j as the sum of currents flowing to node j through current sources:

$$G_j := \sum_{k:(j,k) \in \mathcal{B}_{\text{R}}} g_{jk}, \qquad I_j := \sum_{k:(j,k) \in \mathcal{B}_{\text{CS}}} i_{jk}^{\text{CS}}. \tag{59}$$

Then, we perform a change of variables:

$$s_j := \sqrt{G_j} v_j, \qquad w_{jk} := \frac{g_{jk}}{\sqrt{G_j G_k}}, \qquad b_j := \frac{I_j}{\sqrt{G_j}}, \tag{60}$$

so the energy function of the resistive network rewrites in terms of the new variables:

$$E_{\text{resistive}} = \frac{1}{2} \sum_j s_j^2 - \sum_{j,k} w_{jk} s_j s_k - \sum_k b_k s_k. \tag{61}$$

This is the energy function of an ideal CHN (Eq. (56)). Moreover, the constraints on the electrical potentials (v_k) imposed by the diodes and voltage sources (Eq. (4)) translate into equality and inequality constraints on the new variables (s_k), similar (though not identical) to those of the feasible set of the CHN (Eq. (57)). This shows that (ideal) nonlinear resistive networks are, in some sense, a subclass of (ideal) CHNs.

The converse statement is not true: a CHN with a non-convex energy function cannot be converted into an electrical network composed solely of resistors, diodes, voltage sources and current sources. Such electrical networks have a convex energy function (Theorem 1).

E.3. Comparison of DRNs and DHNs in Simulations

Finally, we compare DRNs and deep Hopfield networks (DHNs) of the same size through simulations. Using equilibrium propagation, we train DHNs of one, two and three hidden layers (each comprising 1024 units), denoted DHN-1H, DHN-2H and DHN-3H, respectively, on the MNIST dataset. Similar to the DRN simulations in Table 1, we use exact block coordinate descent to compute the steady state of the DHN. This contrasts with the DHN simulations of Scellier & Bengio (2017) where the steady state was obtained using gradient descent in the feasible set.

Table 3. Comparison of deep Hopfield network (DHN) and deep resistive network (DRN) architectures on MNIST using equilibrium propagation (EP) for training. Test error rates (in %) are reported as mean ± standard deviation over five runs. We also report the number of iterations of exact block coordinate descent to compute the free state (T) and the nudge state (K).

	DRN			DHN		
	T	K	TEST ERROR (%)	T	K	TEST ERROR (%)
1H	4	4	1.57 ± 0.07	15	15	1.79 ± 0.06
2H	5	5	1.48 ± 0.05	50	20	1.65 ± 0.05
3H	6	6	1.66 ± 0.09	100	10	1.65 ± 0.02

Table 3 shows the results. We observe that DRNs require fewer iterations of exact block coordinate descent than their equivalent-size DHNs to converge to a steady state. This can be explained by their convex energy function, while DHNs may have non-convex energy functions. We also observe that DRNs achieve similar (or even slightly better) performance compared to DHNs. This indicates that the convexity of the energy function in DRNs does not limit their computational expressivity. In fact, Scellier & Mishra (2023) have proven a universal approximation theorem for DRNs.

F. Equilibrium Propagation with Inequality Constraints

The equilibrium propagation (EP) formulas detailed in Appendix B.2 were proved in Scellier & Bengio (2017); Scellier et al. (2024). These derivations assumed that the equilibrium state $s_\star$ satisfies the stationary condition $\frac{\partial E}{\partial s}(s_\star) = 0$. However, this condition may not hold when the feasible set is defined by inequality constraints, as seen in the settings of nonlinear resistive networks (Section 2) and continuous Hopfield networks (Appendix E). In these settings, the stationary condition is satisfied when the equilibrium state $s_\star$ is within the interior of the feasible region $\mathcal{S}$, but it may not hold when $s_\star$ lies on the boundary of $\mathcal{S}$. In this appendix, we demonstrate that the EP formulas remain valid even when the feasible set is defined by inequality constraints.

Let $\Theta := \mathbb{R}^m$ denote the parameter space, and

$$\mathcal{S} := \{s \in \mathbb{R}^n \mid f_1(s) \leq 0, \dots, f_K(s) \leq 0\} \tag{62}$$

denote the feasible set, where $f_k : \mathbb{R}^n \to \mathbb{R}$ is a smooth function for each k. Given two functionals $C : \Theta \times \mathcal{S} \to \mathbb{R}$ and $E : \Theta \times \mathcal{S} \to \mathbb{R}$, called 'cost function' and 'energy function', respectively, we consider the following bilevel optimization problem (Zucchet & Sacramento, 2022):

$$\text{find } \min_{\theta \in \Theta} C(\theta, s(\theta)), \tag{63}$$

$$\text{subject to } s(\theta) := \underset{s \in \mathcal{S}}{\arg\min}\ E(\theta, s). \tag{64}$$

Using the notations from Appendix B, we further define

$$F(\beta, \theta, s) := E(\theta, s) + \beta C(\theta, s), \tag{65}$$

$$s_\theta^\beta := \underset{s \in \mathcal{S}}{\arg\min}\ F(\beta, \theta, s), \tag{66}$$

$$G(\beta, \theta) := F\left(\beta, \theta, s_\theta^\beta\right), \tag{67}$$

$$\mathcal{L}_\beta(\theta) := \frac{G(\beta, \theta) - G(0, \theta)}{\beta}. \tag{68}$$

We refer to $\mathcal{L}_\beta$ as the contrastive function.

Theorem 3 (Equilibrium propagation formulas)**.** *The gradient of the contrastive function $\mathcal{L}_\beta$ is given by:*

$$\nabla_\theta \mathcal{L}_\beta(\theta) = \frac{1}{\beta}\left(\frac{\partial F}{\partial \theta}\left(\beta, \theta, s_\theta^\beta\right) - \frac{\partial F}{\partial \theta}\left(0, \theta, s^0\right)\right). \tag{69}$$

Furthermore, the contrastive function satisfies:

$$\forall \beta \in \mathbb{R}, \quad \mathcal{L}_\beta(\theta) = C\left(\theta, s(\theta)\right) + O(\beta), \qquad \textit{and} \qquad \forall \beta > 0, \quad \mathcal{L}_\beta(\theta) \leq C\left(\theta, s(\theta)\right) \leq \mathcal{L}_{-\beta}(\theta). \tag{70}$$

We prove Theorem 3 by extending the proofs from Scellier et al. (2022; 2024). The new addition is the introduction of the Lagrangian L and the Lagrange multipliers associated with the constrained minimization problems of Eq. (64) and Eq. (66), which brings us back to the setting where the stationary condition is satisfied.

Proof of Theorem 3. Since $\mathcal{S}$ is defined by inequality constraints, Eq. (66) represents constrained optimization problem. We introduce the corresponding Lagrangian function:

$$L\left(\beta, \theta, s, \lambda\right) := F(\beta, \theta, s) + \sum_k \lambda_k f_k(s), \tag{71}$$

where the functions f_k define the feasible set (Eq. (62)), and $\lambda = (\lambda_1, \cdots, \lambda_K) \in \mathbb{R}^K$ are the Lagrange multipliers. Since s_θ^β is a minimum of this constrained optimization problem, there exist Lagrange multipliers $\lambda_\theta^\beta = (\lambda_1^\beta, \cdots, \lambda_K^\beta)$ such that $(s_\theta^\beta, \lambda_\theta^\beta)$ satisfies the Karush-Kuhn-Tucker (KKT) conditions. The stationarity condition for the Lagrangian is

$$\frac{\partial L}{\partial s}\left(\beta, \theta, s_\theta^\beta, \lambda_\theta^\beta\right) = 0, \tag{72}$$

while the primal feasibility condition, dual feasibility condition, and the complementary slackness for inequality constraints are:

$$f_k(s_\theta^\beta) \le 0, \qquad \lambda_k^\beta \ge 0, \qquad \lambda_k^\beta f_k(s_\theta^\beta) = 0, \qquad 1 \le k \le K. \tag{73}$$

Next, we calculate the gradient of the contrastive function:

$$\nabla_\theta \mathcal{L}_\beta(\theta) = \frac{1}{\beta}\left(\frac{\partial G}{\partial \theta}(\beta,\theta) - \frac{\partial G}{\partial \theta}(0,\theta)\right). \tag{74}$$

Using the definitions of G and L, and the complementary slackness condition, we have

$$G(\beta,\theta) = L\left(\beta,\theta,s_\theta^\beta,\lambda_\theta^\beta\right). \tag{75}$$

Therefore, applying the chain rule of differentiation, we get

$$\frac{\partial G}{\partial \theta}(\beta,\theta) = \frac{\partial L}{\partial \theta}\left(\beta,\theta,s_\theta^\beta,\lambda_\theta^\beta\right) + \underbrace{\frac{\partial L}{\partial s}\left(\beta,\theta,s_\theta^\beta,\lambda_\theta^\beta\right)}_{=0}\cdot\frac{\partial s_\theta^\beta}{\partial \theta} + \sum_k \frac{\partial L}{\partial \lambda_k}\left(\beta,\theta,s_\theta^\beta,\lambda_\theta^\beta\right)\cdot\frac{\partial \lambda_k^\beta}{\partial \theta} \tag{76}$$

$$= \frac{\partial F}{\partial \theta}\left(\beta,\theta,s_\theta^\beta\right) + \sum_k f_k(s_\theta^\beta)\cdot\frac{\partial \lambda_k^\beta}{\partial \theta} \tag{77}$$

Here we have used the stationarity condition of the Lagrangian, Eq. (72). We claim that

$$f_k(s_\theta^\beta)\cdot\frac{\partial \lambda_k^\beta}{\partial \theta} = 0, \qquad 1 \le k \le K, \tag{78}$$

which we prove next. Let k be fixed. The primal feasibility condition of Eq. (73) imposes that $f_k(s_\theta^\beta) \le 0$. If $f_k(s_\theta^\beta) = 0$, then Eq. (78) holds. Otherwise, $f_k(s_\theta^\beta) < 0$. Using the smoothness of f_k and the smoothness of $\theta' \mapsto s_{\theta'}^\beta$ (guaranteed by the implicit function theorem, assuming that F is smooth too), there exists a neighborhood of θ such that $f_k(s_{\theta'}^\beta) < 0$ for every θ' in this neighborhood. By the complementary slackness of Eq. (73), it follows that $\lambda_k^\beta = 0$ for every θ' in this neighborhood, therefore $\frac{\partial \lambda_k^\beta}{\partial \theta} = 0$. Hence, Eq. (78) holds again. As a consequence

$$\frac{\partial G}{\partial \theta}(\beta,\theta) = \frac{\partial F}{\partial \theta}\left(\beta,\theta,s_\theta^\beta\right). \tag{79}$$

Hence,

$$\nabla_\theta \mathcal{L}_\beta(\theta) = \frac{1}{\beta}\left(\frac{\partial F}{\partial \theta}\left(\beta,\theta,s_\theta^\beta\right) - \frac{\partial F}{\partial \theta}\left(0,\theta,s_\theta^0\right)\right), \tag{80}$$

which is the first half of Theorem 3.

Next, we prove the second half: the properties of the contrastive function. Using again the chain rule of differentiation, we have

$$\frac{\partial G}{\partial \beta}(\beta,\theta) = \frac{\partial L}{\partial \beta}\left(\beta,\theta,s_\theta^\beta,\lambda_\theta^\beta\right) + \underbrace{\frac{\partial L}{\partial s}\left(\beta,\theta,s_\theta^\beta,\lambda_\theta^\beta\right)}_{=0}\cdot\frac{\partial s_\theta^\beta}{\partial \beta} + \sum_k \frac{\partial L}{\partial \lambda_k}\left(\beta,\theta,s_\theta^\beta,\lambda_\theta^\beta\right)\cdot\frac{\partial \lambda_k^\beta}{\partial \beta} \tag{81}$$

$$= C\left(\theta,s_\theta^\beta\right) + \sum_k \underbrace{f_k(s_\theta^\beta)\cdot\frac{\partial \lambda_k^\beta}{\partial \beta}}_{=0} = C\left(\theta,s_\theta^\beta\right). \tag{82}$$

Here again, we have used the stationary condition of the Lagrangian, and the fact that

$$f_k(s_\theta^\beta)\frac{\partial \lambda_k^\beta}{\partial \beta} = 0, \qquad 1 \le k \le K, \tag{83}$$

which we prove similarly as before. Evaluating the above expression at $\beta = 0$, we get

$$\frac{\partial G}{\partial \beta}(0, \theta) = C(\theta, s(\theta)), \tag{84}$$

since $s_\theta^0 = s(\theta)$ by definition. Using a Taylor expansion of $G(\beta, \theta)$ around $\beta = 0$, we get

$$G(\beta, \theta) = G(0, \theta) + \beta C(\theta, s(\theta)) + O(\beta^2). \tag{85}$$

Subtracting $G(\theta, 0)$ on both sides and dividing by β, we get

$$\mathcal{L}_\beta(\theta) = \frac{G(\beta, \theta) - G(0, \theta)}{\beta} = C(\theta, s(\theta)) + O(\beta). \tag{86}$$

As for the upper bound and lower bound properties, we write for any $\beta \in \mathbb{R}$,

$$G(\beta, \theta) = \inf_{s \in \mathcal{S}} \{E(\theta, s) + \beta C(\theta, s)\} \leq E(\theta, s(\theta)) + \beta C(\theta, s(\theta)) = G(0, \theta) + \beta C(\theta, s(\theta)). \tag{87}$$

Subtracting $G(0, \theta)$ on both sides we get

$$G(\beta, \theta) - G(0, \theta) \leq \beta C(\theta, s(\theta)). \tag{88}$$

Next we divide by β. For $\beta > 0$, we get

$$\mathcal{L}_\beta(\theta) = \frac{G(\beta, \theta) - G(0, \theta)}{\beta} \leq C(\theta, s(\theta)), \qquad \beta > 0, \tag{89}$$

and for $\beta < 0$ we get

$$\mathcal{L}_\beta(\theta) = \frac{G(\beta, \theta) - G(0, \theta)}{\beta} \geq C(\theta, s(\theta)), \qquad \beta < 0. \tag{90}$$

Hence the result. □