Nonlinear Classification Without a Processor

Sam Dillavou
Department of Physics and Astronomy
University of Pennsylvania
dillavou@sas.upenn.edu

Benjamin D. Beyer
Department of Physics and Astronomy
University of Pennsylvania

Menachem Stern
Department of Physics and Astronomy
University of Pennsylvania

Marc Z. Miskin
Department of Electrical and Systems Engineering
University of Pennsylvania

Andrea J. Liu
Department of Physics and Astronomy
University of Pennsylvania

Douglas J. Durian
Department of Physics and Astronomy
University of Pennsylvania

Abstract

Computers, as well as most neuromorphic hardware systems, use central processing and top-down algorithmic control to train for machine learning tasks. In contrast, brains are ensembles of 100 billion neurons working in tandem, giving them tremendous advantages in power efficiency and speed. Many physical systems 'learn' through history dependence, but training a physical system to perform arbitrary nonlinear tasks without a processor has not been possible. Here we demonstrate the successful implementation of such a system - a learning meta-material. This nonlinear analog circuit is comprised of identical copies of a single simple element, each following the same local update rule. By applying voltages to our system (inputs), inference is performed by physics in microseconds. When labels are properly enforced (also via voltages), the system's internal state evolves in time, approximating gradient descent. Our system *learns on its own*; it requires no processor. Once trained, it performs inference passively, requiring approximately 100 μW of total power dissipation across its edges. We demonstrate the flexibility and power efficiency of our system by solving nonlinear 2D classification tasks. Learning meta-materials have immense potential as fast, efficient, robust learning systems for edge computing, from smart sensors to medical devices to robotic control.

1 Introduction

The brain is the ultimate meta-material. Comprised of 100 billion neurons signaling and evolving in tandem, its emergent behaviors include physical locomotion, learning, memory, coding, and writing Computer Science manuscripts. Computers, in contrast, operate on a hierarchical basis, with processors performing centralized computation, shoehorned into digital logic [1]. While artificial systems are catching up to human brains in many respects, they possess this distinct structural disadvantage, and are thus are less power-efficient [2, 3] and slower at complex tasks like object recognition [4], especially once their factor of 10^6 advantage in signal timescale (ns vs ms) is taken into consideration.

The field of neuromorphic computing seeks to shrink these gaps between computers and brains by building hardware that is, at least in spirit, more 'brain-like'. Typically these efforts involve

37th First Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2023(MLNPCP 2023).

integrating memory and computation, or allowing physical processes to perform analog computations for inference [1, 3, 5, 6, 7, 8, 9, 10, 11, 12]. However, these tactics retain a central design aspect absent in the brain – hierarchical, top-down control during learning.

A nascent field, physical learning [13], seeks to create physical systems whose structures channel physics into performing useful tasks, without encoding and decoding signals to and from digital logic. For a limited set of tasks and conditions, this concept has been achieved using common physical systems such as elastic foams [14], and in principle any physical system with history dependence can be 'trained,' *e.g.* polymer glasses or frictional interfaces [15, 16]. Designed systems have achieved a much wider range of functionalities, but are typically trained with computational assistance (that is, in a 'top-down' manner) [17, 18, 14, 19, 20, 21], through an iterative process, similar to artificial neural networks (ANNs).

Recently, the overlap of these two fields has produced 'learning meta-materials,' analog electrical networks that evolve in a bottom-up manner like a physical system, but which, like an ANN, can be trained to perform arbitrary functionality by example [22, 23, 24]. These systems are built to implement Coupled Learning (CpL) [25], a framework inspired by Contrastive Hebbian Learning (CHL) [26] and Equilibrium Propagation (EP) [27, 28], in which learning is driven by comparison of two states of the same system. These first-generation learning meta-materials can perform a variety of tasks, including regression and classification, but their functionality is limited by the linearity of their elements, and the discrete values of their adjustable parameters.

Here we demonstrate nonlinear classification in a second-generation learning meta-material, a standalone physical system that can be trained by example, like an artificial neural network (ANN) [29]. Our system, an analog electrical network of MOSFET transistors, is comprised of 32 identical copies of a single self-adjusting element. Training examples are shown to the network as enforced voltages (boundary conditions), and the collective evolution of each element generates supervised learning as an emergent property, requiring no outside computation. Once trained, the system is a passive nonlinear electrical network, and inference is performed by physics in microseconds dissipating approximately 100 micro Watts of power across its edges. We demonstrate the flexibility and efficacy of our system by training it to perform several linear and nonlinear 2D classification tasks.

2 Learning Meta-Material Operation

Our system can be trained in a manner similar to an artificial neural network, that is, iteratively and by example (supervised learning). However, it requires no processor, and its inner workings are far closer to physical systems with history-dependent evolution in time [14, 15, 16]. This combination is achieved by implementing the Coupled Learning (CpL) framework, as follows.

Consider an electrical network comprised of edges with conductance $\vec{K}$. Applying 'input' voltages $\vec{I}$ to two or more nodes of the network will generate a new electrical state inside the system (currents will flow). Picking two arbitrary nodes and defining the difference between them as the 'output' $O \equiv O_+ - O_-$ (as in [28]), we may think of the network as guiding physics to perform a linear function, $F(\vec{I})_{\vec{K}} = O$. By modifying $\vec{K}$, a wide range of linear functions may be created.

In this work, the edges of our network are N-channel enhancement-mode MOSFET transistors wired as nonlinear variable resistors. Specifically, we utilize these transistors in the Ohmic (passive) regime, where their source-drain conductance follows $K \propto G - V_{th} - \overline{V}$, where G is the gate voltage, $V_{th} \approx 0.6$ V is a fixed, threshold voltage, and $\overline{V}$ is the average of the source and drain voltages. This last contribution is a source of nonlinearity; the conductance of each edge depends on the electrical state of the system, and thus the inputs themselves. We may now think of the physics acting on the network as performing a nonlinear function

$$\mathcal{N}(\vec{I})_{\vec{G}} = O \tag{1}$$

The 'learning' in our system occurs by changing the gate voltages $\vec{G}$ of these transistor edges. We utilize the learning rule from CpL, which requires comparing two states of the same system, *free* and *clamped*. In the *free* state, only inputs are applied, and the output is determined by Eq. 1. In the *clamped* state, inputs are applied, but the output is also enforced, specifically at a value between the free output O and the desired output (label) L

$$O^C = \eta L + (1 - \eta) O \tag{2}$$

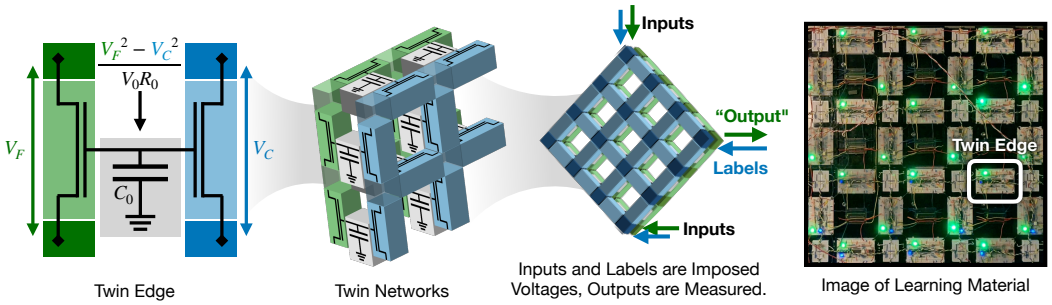

Figure 1: **Structure of the Learning Meta-Material.** This prototype is comprised of 32 identical copies of a single self-adjusting element, shown schematically on the left. This element, a *twin-edge*, consists of two N-channel MOSFETs whose gates are tied to a shared capacitor. This capacitor is charged/discharged by local circuitry that produces current according to the learning rule (Eq. 3). These elements are wired together such that they form identical *twin networks* that can be viewed as two copies of the same nonlinear electrical network. This system can then be trained by imposing inputs to both networks and labels to just one, creating two electrical states of the 'same' system. The learning rule embedded on each element will evolve according to differences between the networks, and collectively produce supervised learning as an emergent phenomenon. The physical circuit (right side), shown here without periodic boundary conditions, is a standalone meta-material, requiring no processor to learn or perform inference.

where $\eta \leq 1$ is a hyper-parameter. In practice, instead of clamping the difference between the two output nodes $O^C = O^C_+ - O^C_-$, we clamp each output node individually as $O^C_\pm = O_\pm \pm \frac{\eta}{2}(L - O)$. Each gate voltages of our system $\vec{G}$ then evolves according to the CpL learning rule, specifically,

$$\dot{G} = \frac{V_F^2 - V_C^2}{V_0 R_0 C_0} \tag{3}$$

Where V_0, R_0, and C_0 are constants set by circuitry components, and V_F and V_C are the voltage drops across an edge of the network in the free and clamped states respectively.

As in previous work [22], we build a twin network such that we may simultaneously access the free and clamped states. These twin networks are constructed from a single irreducible element, a twin edge, shown in Fig. 1. These edges do not interact directly, but share a common gate voltage to ensure commensurate edges of each network are identical. Circuitry on board each twin edge operationalizes the learning rule (Eq. 3) by charging a shared gate capacitor with capacitance C_0. This learning may be frozen/unfrozen via a global binary switch. When the system is frozen, the charged capacitors are connected only to the transistor gates, and thus their charge decays slowly but has a finite lifetime. After training, testing the circuit requires no more than 100ms, and we find no appreciable affect of this decay over that timescale. For long-term storage of trained values, nonvolatile memory such as memristors may be necessary.

An ensemble of 32 of these twin edges comprise our system, to which we apply boundary conditions (Eq. 2), driving evolution (Eq. 3). Our system, shown in Fig. 1, right side, is a standalone, active meta-material. The role of a supervisor in its training is therefore quite simple. For a chosen data point i:

1. Enforce the inputs $\vec{I}_i$ on both networks, physics produces output O_i (Eq. 1)
2. Enforce the clamped output O^C_i (Eq. 2).
3. Unfreeze the learning for training step time t_h, allowing edges to evolve (Eq. 3)

At no point does the supervisor calculate a solution or an update, load or save weights, or communicate with the edges in any way other than the binary 'freeze' switch. Further, in practice, step 2 is performed automatically by a feedback circuit that continually enforces Eq. 2. In this way, evolution is truly continuous, as modifications to $\vec{G}$ impact O and thus O^C at a timescale much faster than learning, $\tau_V \approx 1$ μs, the measured equilibration timescale of the network. The learning evolution (Eq. 3) has

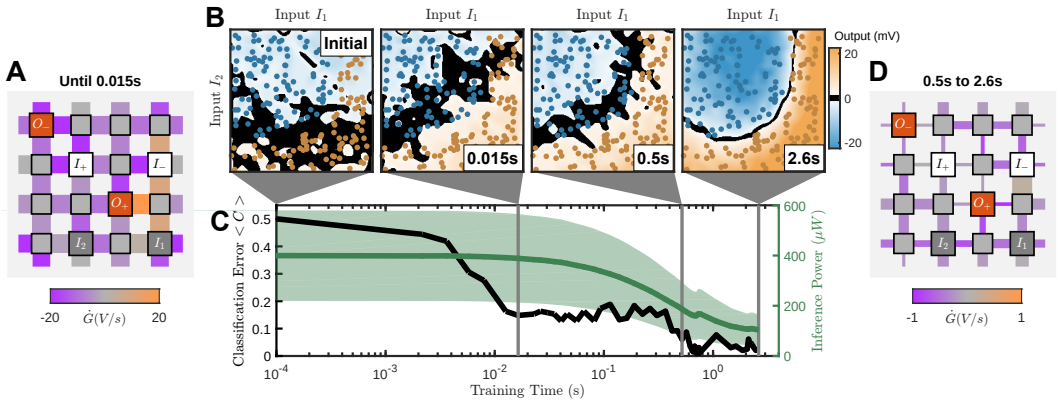

Figure 2: **A 2D Nonlinear Classification Task** (A) Schematic of the network showing input and output nodes as well as average rate of change in learning degrees of freedom for the first 15 ms of training. Edge width indicates average edge conductance across the entire training set. (B) Classification task visualization at four time points during training. Range of both axes is 0 to 0.45V, the entire range of allowed node voltages. The two classes of training data are overlaid as blue and orange colored dots, and the system output $O(t) = O_+(t) - O_-(t)$ as a function of inputs (I_1, I_2) is the background color. The black region indicates where $|O| < 1.4$ mV, near enough to the boundary that noise potentially interferes with evaluation. (C) Classification error (black) and average power dissipated across edges in the inference state from entire input range (green) versus time. Faded green indicates standard deviation. Time points indicated in (B) are shown as vertical gray bars. (D) Network schematic as in (A) but at the end of training. Colors indicate average change in learning degrees of freedom from 0.5 to 2.6 seconds of training.

approximate timescale $\tau_0 \approx 24$ ms. Cycling through data points in this three step manner will train the system, just like an ANN performing supervised learning.

Because we are interested in performing classification tasks, we add one additional feature to our learning protocol, designed to maximize classification accuracy instead of minimizing mean squared error. Our classification tasks will each contain two classes, and we will treat zero output ($O_+ = O_-$) as the classification boundary. Therefore our classification error for data point i can be written as

$$C_i = \Theta(-L_i O_i) \tag{4}$$

Where Θ is the Heaviside step function, O_i is the output for data point i, and label L_i takes a single positive value for all members of class one and the negative of that value for all members of class two. This results in $C_i = 0$ for correctly classified data points, and $C_i = 1$ for incorrect ones. Then, for each training step, we add a caveat to step 3: evolve *only* if the chosen data point is *incorrectly* classified[1]. Note that this is an additional task for the supervisor, but involves only a binary evaluation. In practice, noise in the system will make for occasional incorrect evaluations of points very near the boundary, which will result in a natural push towards widening the gap between classes, and more consistent evaluations. In this way, small-scale noise becomes a robustness-building feature of our system. Finally, while we do not include such tasks in this work, classification with more than two classes may be accomplished by adding additional output nodes, as previously demonstrated in linear learning metamaterials [22].

3 Results

Using the protocol detailed above, we perform nonlinear classification tasks using our learning meta-material. We use the same network setup for each task, shown in Fig. 2A. We impose two constant inputs $I_- = 0.11V$ and $I_+ = 0.33V$, which remain fixed for all tasks shown. We also

[1]In practice it speeds training to store evaluations of a data point and update or not update based on one evaluation several times. We find this speeds training without an appreciable decrease in accuracy, but is not necessary for successful functioning. More sophisticated evaluation methods (*e.g.* not updating a data point that is very far correctly classified for a longer period) would likely speed training even further

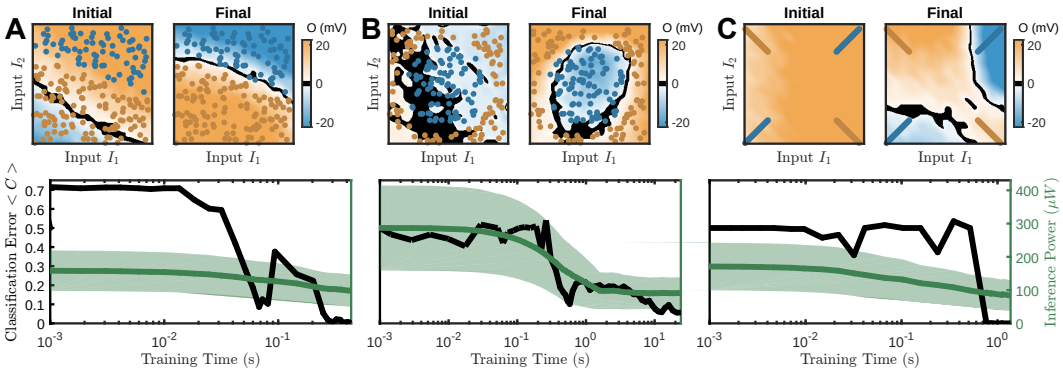

Figure 3: **Additional Classification Tasks** (A) Linear task. Top: Output maps. Orange and blue dots are training data of class 1 and 2 respectively, background color is network output O, and black indicates $|O| < 1.4$ mV. The maps are initial and final inference. Bottom: classification accuracy (black) and average power dissipation over all edges during inference (green), over training time. (B) and (C) are the same as (A) but for a closed circular boundary (B) and a task that emulates XOR (C). Note that for (C) the training data points were not randomly chosen but rather 196 data points are spaced equally in four lines, as shown. Variation in initial inference power between (A-C) comes from variation in initial gate voltage G_0 values.

impose two variable inputs I_1 and I_2, which vary between 0 and $0.45V$, and utilize the differential output scheme $O = O_+ - O_-$ as previously described. We randomly (with uniform probability) select 196 values of I_1 and I_2 from voltages allowed in our network [0, 0.45]V. We choose a 2D boundary, and re-select all data points that lie within a small distance ($\approx 0.01V$) in (I_1,I_2) space. We choose label values $L_1 = -L_2$ for the two classes. G are initialized at an approximately uniform value in the middle of their range.

We choose a highly nonlinear boundary (a circular arc) to define our first task, and successfully train the network, as shown by the output maps in Fig. 2B. The network trains (evolves G) for less than three total seconds, with peak accuracy above 99% and final accuracy 96.9%. As the majority of G values decrease during training, the power to perform inference falls from approximately 400 μW to a final average of 105 ± 62 μW. As the network equilibrates in approximately 2 μsec, this represents an inference energy of 210 pJ, or 7 pJ per parameter (edge). This number is a summation of power dissipation across every edge in the equilibrium state. This calculation ignores a number of factors, including capacitance of the implementation (breadboards and wires) as well as of the transistors themselves. By the end of training, the network evolution has massively slowed, as shown in Fig. 2D, and the decision boundary hovers and wiggles around the optimal line due to small-scale noise in evaluation.

We perform a similar training protocol for three additional tasks, shown in Fig. 3A, B, and C. These tasks all end their training above 94% accuracy, with a comparable inference energy to the task in Fig. 2. Note that the task in 3A is linear, and the task in 3C does not have random but rather prescribed input values (I_1, I_2), and is meant to emulate XOR. The variety of these tasks, all with the same input/output node selection, highlight the flexibility and power-efficiency of our system. While it is significantly faster to learn simpler (linear) tasks, determining the minimum network size and connectivity needed to perform tasks of varying complexity is a subject for future work. Details and results for all four tasks, including hyper-parameters, training times, and power/energy for inference, are listed in Table 1.

4 Conclusion

We have demonstrated the efficacy, flexibility, speed, and energy efficiency of our nonlinear learning meta-material. Our system trains itself, with each element self-adjusting in response to imposed boundary conditions (training data). Using one input/output scheme and a variety of hyper-parameters, our network learned four 2D classification tasks, three of them nonlinear. After training, a single evaluation of each task dissipates 7 pJ per edge (parameter) or less on average. It should be noted

Table 1: Task Details and Results

Fig	η	G_0	$\lvert L\rvert$	Tr. Time	Epochs	Acc.	Inference Pwr	Inf En	pJ/Param
2	0.62	4.6V	0.18V	2.6 sec	12755	96.9%	105$\pm$62 μW	210 pJ	7
3A	1	2.3V	0.09V	0.44 sec	797	100%	99$\pm$48 μW	198 pJ	6
3B	0.62	3.5V	0.18V	24 sec	12755	94.3%	91$\pm$48 μW	182 pJ	6
3C	0.5	2.3V	0.18V	1.4 sec	15944	100%	84$\pm$46 μW	167 pJ	5

Hyperparameters and results for each experiment. Hyper parameters η (nudge factor), initial value for all learning degrees of freedom G_0, label magnitude $|L|$, and training epochs are determined at the start of each experiment to demonstrate a range of successful values. Epochs were chosen to be significantly larger than required, to demonstrate the stability of the final solutions. Training time represents integrated time spent evolving the system, which only occurs for incorrectly classified data points; when accuracy is higher, this time is reduced per epoch. Results accuracy (Acc.), inference power, inference energy, and pJ/param are calculated using the final learned state of the system. Inference power includes only energy dissipated by effective resistance of network edges. Inference energy (Inf En) is estimated using (inference power * 2 μsec). All power and energy values are reported using the average across the training set. pJ/param is (inference energy / 32).

that this is a lower bound for power consumption since it ignores several relevant factors including power loss from parasitic capacitance from the breadboards and transistors. However, we find it is a useful benchmark, as it illustrates the efficiency of the analog computation used in our method. Like all systems that integrate computation with memory, there is no need to spend additional energy shuffling information (e.g. weight values) in and out of the network. As a point of comparison, the most efficient supercomputer on the Green500 list[2] consumes approximately 15 pJ per FLOP (floating point operation)[3], and dense neural networks perform inference using approximately 1 FLOP/parameter. Our parameters G and our updates are orders of magnitude less precise than they would be using simulated networks and floating point numbers, but for tasks requiring only modest precision, it is easy to see how future versions of network may be competitive.

Learning meta-materials show great promise as fast, energy-efficient *in-situ* learning devices, with applications in edge computation, medicine, agriculture, robotics, and more. Furthermore, because updates are performed on each element with physics performing the inference calculation, the system is tolerant to damage/defects like its first-generation cousins [22]. Discrepancies between the twin networks is a potential source of error, and subject for future work. However, because updates are local, errors due to physical imperfection do not compound in the same manner as in differentiation (backpropagation) of physical networks, and the system is insensitive to variations between transistors within each network. As a result, learning meta-materials should be massively scalable, and may one day prove competitive for machine learning applications.

Broader Impact

Computational machine learning has become widespread and energy use is increasing at an unsustainable rate. There is therefore both a need for more efficient computation at a large (supercomputer) and small (edge computing) scale. Neuromorphic hardware seeks to lower the energy cost of machine learning applications. The system demonstrated here is energy-efficient and flexible, and may prove useful for compact, low-power edge computation, decreasing data transfer and storage energy costs. Furthermore, like its first-generation counterparts [22], it is robust to damage, making it a potentially useful system to combat electronic waste, and potentially manufacturable at scale.

Acknowledgments and Disclosure of Funding

This research was supported by the National Science Foundation via the UPenn MRSEC/DMR-1720530 and MRSEC/DMR-DMR-2309043 (S.D. and D.J.D.), and DMR-2005749 (A.J.L.), the Simons Foundation # 327939 (A.J.L.), and the U.S. Department of Energy, Office of Basic Energy Sciences, Division of Materials Sciences and Engineering award DE-SC0020963 (M.S.). S.D.

[2] `https://www.top500.org/lists/green500/2023/06/`

[3] 65.396 GigaFLOPS/Watt = 65.396 GigaFLOP/Joule $\rightarrow$ 15.38 pJ per FLOP

acknowledges support from the University of Pennsylvania School of Arts and Sciences Data Driven Discovery Initiative. M.Z.M. acknowledges support from the Packard Foundation. D.J.D. and A.J.L. thank CCB at the Flatiron Institute, as well as the Isaac Newton Institute for Mathematical Sciences under the program "New Statistical Physics in Living Matter" (EPSRC grant EP/R014601/1), for support and hospitality while a portion of this research was carried out.

References

[1] C. Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, October 1990.

[2] Vijay Balasubramanian. Brain power. *Proceedings of the National Academy of Sciences*, 118(32):e2107022118, August 2021.

[3] Dennis V. Christensen, Regina Dittmann, Bernabé Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, Ilia Valov, Gianluca Milano, Carlo Ricciardi, Shi-Jun Liang, Feng Miao, Mario Lanza, Tyler J. Quill, Scott T. Keene, Alberto Salleo, Julie Grollier, Danijela Marković, Alice Mizrahi, Peng Yao, J. Joshua Yang, Giacomo Indiveri, John Paul Strachan, Suman Datta, Elisa Vianello, Alexandre Valentian, Johannes Feldmann, Xuan Li, Wolfram H. P. Pernice, Harish Bhaskaran, Steve Furber, Emre Neftci, Franz Scherr, Wolfgang Maass, Srikanth Ramaswamy, Jonathan Tapson, Priyadarshini Panda, Youngeun Kim, Gouhei Tanaka, Simon Thorpe, Chiara Bartolozzi, Thomas A. Cleland, Christoph Posch, Shih-Chii Liu, Gabriella Panuccio, Mufti Mahmud, Arnab Neelim Mazumder, Morteza Hosseini, Tinoosh Mohsenin, Elisa Donati, Silvia Tolu, Roberto Galeazzi, Martin Ejsing Christensen, Sune Holm, Daniele Ielmini, and N. Pryds. 2022 Roadmap on Neuromorphic Computing and Engineering. *arXiv:2105.05956 [cond-mat]*, October 2021.

[4] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, July 2022.

[5] Rui Wang, Tuo Shi, Xumeng Zhang, Jinsong Wei, Jian Lu, Jiaxue Zhu, Zuheng Wu, Qi Liu, and Ming Liu. Implementing in-situ self-organizing maps with memristor crossbar arrays for data mining and optimization. *Nature Communications*, 13(1):2289, April 2022.

[6] Peng Yao, Huaqiang Wu, Bin Gao, Jianshi Tang, Qingtian Zhang, Wenqiang Zhang, J. Joshua Yang, and He Qian. Fully hardware-implemented memristor convolutional neural network. *Nature*, 577(7792):641–646, January 2020.

[7] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams. Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2016.

[8] Logan G. Wright, Tatsuhiro Onodera, Martin M. Stein, Tianyu Wang, Darren T. Schachter, Zoey Hu, and Peter L. McMahon. Deep physical neural networks trained with backpropagation. *Nature*, 601(7894):549–555, January 2022.

[9] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. Survey of Machine Learning Accelerators. In *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–12, September 2020.

[10] Sungho Kim, Chao Du, Patrick Sheridan, Wen Ma, ShinHyun Choi, and Wei D. Lu. Experimental Demonstration of a Second-Order Memristor and Its Ability to Biorealistically Implement Synaptic Plasticity. *Nano Letters*, 15(3):2203–2211, March 2015.

[11] Jérémie Laydevant, Danijela Markovic, and Julie Grollier. Training an Ising Machine with Equilibrium Propagation, May 2023.

[12] Tie Mei and Chang Qing Chen. In-memory mechanical computing. *Nature Communications*, 14(1):5204, August 2023.

[13] Menachem Stern and Arvind Murugan. Learning Without Neurons in Physical Systems | Annual Review of Condensed Matter Physics. https://www-annualreviews-org.proxy.library.upenn.edu/doi/full/10.1146/annurev-conmatphys-040821-113439.

[14] Nidhi Pashine, Daniel Hexner, Andrea J. Liu, and Sidney R. Nagel. Directed aging, memory, and nature's greed. *Science Advances*, 5(12):4215, December 2019.

[15] A. J. Kovacs, J. J. Aklonis, J. M. Hutchinson, and A. R. Ramos. Isobaric volume and enthalpy recovery of glasses. II. A transparent multiparameter theory. *Journal of Polymer Science: Polymer Physics Edition*, 17(7):1097–1162, 1979.

[16] Sam Dillavou and Shmuel M. Rubinstein. Nonmonotonic Aging and Memory in a Frictional Interface. *Physical Review Letters*, 120(22):224101, June 2018.

[17] Jason W. Rocks, Nidhi Pashine, Irmgard Bischofberger, Carl P. Goodrich, Andrea J. Liu, and Sidney R. Nagel. Designing allostery-inspired response in mechanical networks. *Proceedings of the National Academy of Sciences*, 114(10):2520–2525, March 2017.

[18] Jason W. Rocks, Henrik Ronellenfitsch, Andrea J. Liu, Sidney R. Nagel, and Eleni Katifori. Limits of multifunctionality in tunable networks. *Proceedings of the National Academy of Sciences*, 116(7):2506–2511, February 2019.

[19] Nidhi Pashine. Local rules for fabricating allosteric networks. *Physical Review Materials*, 5(6):065607, June 2021.

[20] C. Kaspar, B. J. Ravoo, W. G. van der Wiel, S. V. Wegner, and W. H. P. Pernice. The rise of intelligent matter. *Nature*, 594(7863):345–355, June 2021.

[21] Jonathan B. Hopkins, Ryan H. Lee, and Pietro Sainaghi. Using binary-stiffness beams within mechanical neural-network metamaterials to learn. *Smart Materials and Structures*, 32(3):035015, February 2023.

[22] Sam Dillavou, Menachem Stern, Andrea J. Liu, and Douglas J. Durian. Demonstration of Decentralized Physics-Driven Learning. *Physical Review Applied*, 18(1):014040, July 2022.

[23] J. F. Wycoff, S. Dillavou, M. Stern, A. J. Liu, and D. J. Durian. Desynchronous learning in a physics-driven learning network. *The Journal of Chemical Physics*, 156(14):144903, April 2022.

[24] Menachem Stern, Sam Dillavou, Marc Z. Miskin, Douglas J. Durian, and Andrea J. Liu. Physical learning beyond the quasistatic limit. *Physical Review Research*, 4(2):L022037, May 2022.

[25] Menachem Stern, Daniel Hexner, Jason W. Rocks, and Andrea J. Liu. Supervised Learning in Physical Networks: From Machine Learning to Learning Machines. *Physical Review X*, 11(2):021045, May 2021.

[26] Javier R. Movellan. Contrastive Hebbian Learning in the Continuous Hopfield Model. In David S. Touretzky, Jeffrey L. Elman, Terrence J. Sejnowski, and Geoffrey E. Hinton, editors, *Connectionist Models*, pages 10–17. Morgan Kaufmann, January 1991.

[27] Benjamin Scellier and Yoshua Bengio. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 11, 2017.

[28] Jack Kendall, Ross Pantone, Kalpana Manickavasagam, Yoshua Bengio, and Benjamin Scellier. Training End-to-End Analog Neural Networks with Equilibrium Propagation. *arXiv:2006.01981 [cs]*, June 2020.

[29] Sam Dillavou, Benjamin Beyer, Menachem Stern, Marc Z. Miskin, Andrea J. Liu, and Douglas J. Durian. Circuits that train themselves: Decentralized, physics-driven learning. In *AI and Optical Data Sciences IV*, volume 12438, pages 115–117. SPIE, March 2023.