Local learning with highly analog memory devices

Christopher H. Bennett

université PARIS-SACLAY

NNT : 2018SACLS037

UNIVERSITÉ PARIS SUD
Comprendre le monde, construire l'avenir

Apprentissage local avec des dispositifs de mémoire hautement analogiques

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

École doctorale n°575 Physique et ingénierie : électrons, photons, sciences du vivant (EOBE)
Spécialité de doctorat : électronique et optoélectronique, nano et microtechnologies

Thèse présentée et soutenue à ORSAY, le 8 Février 2018, par

M. Christopher H. BENNETT

Composition du Jury :

Mme. Cristell MANEUX Professeure, IMS, Université Bordeaux	Présidente
M. Marc BOCQUET Maître de conférences, IM2NP, Aix-Marseille Université	Rapporteur
M. Bertrand GRANADO Professeur, LIPAC, Sorbonne Université	Rapporteur
M. Vincent DERYCKE Chercheur, LICSEN, CEA Paris-Saclay	Examinateur
M. Sylvain SAIGHI Maître de conférences, IMS, Université Bordeaux	Examinateur
M. Jacques-Olivier KLEIN Professeur, Université Paris-Sud	Directeur de thèse
M. Damien QUERLIOZ Chargé de recherche, C2N, Orsay	Co-directeur de thèse

Thèse de doctorat

Thèse effectuée au sein du **Centre de Nanosciences et de Nanotechnologies**
de l'Université Paris-Sud
Bâtiment 220, rue André AMPÈRE
91405 ORSAY cedex
FRANCE

To my parents, for always believing in me, even when I forgot to do so myself. . . .
And to Laura, my love and best friend, I'm glad we were there for each other during our PhD-'lives' ;) ! . . .

Acknowledgements

First and foremost, many sincere thanks to Prof. Jacques-Olivier Klein for his excellent guidance and feedback throughout my degree, and giving me the opportunity to launch my research career in Orsay!

An additional massive thanks is due to Dr. Damien Querlioz for co-advising, and for opening the way to amazing projects, ideas, and conferences I may not have considered otherwise!

I would like to thank Prof. Cristell Maneux for serving excellently as the president of the jury, and to my other committee members, Prof. Bertrand Granado, Dr. Marc Bocquet, Dr. Vincent Derycke, and Dr. Sylvain Saighi for your important contributions and questions during the day of my defense. The experience was challenging, and most of all memorable, being held on a rare Parisian snow-day!

Next, I would be remiss not to thank Dr. Yu-Pu Lin , for fabricating the final organic synapse devices we used in our hardware learning experiments, and for inspiring me with your hard work. In this context, I would like to yet again thank Dr. Vincent Derycke, for your excellent advise and guidance on this project, from its origin until our publication.

To Dr. Fabien Alibart and Dr. Selina LaBarbera, for making our collaborations on your filamentary nanodevices so personally warm and academically productive!

To Dr. Laurie Calvet, for your excellent feedback and help with collaborations and extensions of my work, and for always urging me to keep reading!

To Dr. Joseph Friedman, for welcoming me on my first day in Paris-Sud, your useful feedback/ career advice, and being a great companion on runs and as American expats abroad.

To Prof. Sandip Tiwari, his inspiring lectures while attending Paris-Sud pushed me intellectually, and his wise guidance and mentorship in other moments was also critical.

To Dr. Julie Grollier, for organizing so many useful reading groups and the Biocomp events with our labs, all of which were inspiring and useful!

To Prof. Giaocomo Indiveri, Tobi Delbruck, Rodney Douglas, and the other co-organizers of the Capocaccia Neuromorphic Engineering Conference– for organizing a truly special collaborative research experience that I was very fortunate to attend twice!

To Dr. Nicholas Locatelli, for being the kindest and most helpful colleague imaginable during your tenure at Paris-Sud!

Lastly, and certainly not least, a huge thanks to my fellow 'co-doctorants' at Paris-Sud- notably Damir Vodenicarevic, Alice Mizrahi, Adrien Vincent, Maxence Ernoult, and Tifenn Hirtzlin – your feedback, encouragement, humor, and companionship at so many coffee breaks really got me through.

Contents

List of Figures

Introduction

"For a successful technology, reality must take precedence over public relations, for Nature cannot be fooled."

Richard P. Feynmann

"NANOELECTRONICS *, far from being a niche field of material science, may hold a foundational importance to the future of several hundred billion dollar digital computing industries that form the bedrock of modern global communications, information technology (IT), and data storage. In this introductory chapter, global problem context is illustrated for the important task of developing and co-integrating emerging new architectures and nanodevices for reduced energy consumption and improved scaling as billions of new grid-connected devices come online in the next decade.* "

By 2018, projections estimate that internet-connected devices- servers, laptops, phones, and tablets double the number of humans in all- will account for 12% of the world's electricity consumption (2.5 PW) [1]. Somewhat surprisingly, less than a quarter of this total consumption comes from the large server farms so often associated with the IT industry's footprint; in a Berkeley Lab report, this use was estimated at 61 kWh/year, constituting 1.5 % of total US Grid consumption [2]. While the same report identified major opportunities for energy efficiency and reducing power use at scale, peripheral, e.g. home or mobile use energy costs have continued to increase super-linearly [1]. The International Energy Agency confirms this trend in a 2009 report which notes that "the growth of electricity consumption by *small* electrical and electronic devices has been the most rapid of all appliance categories over the past five years in both OECD and non-OECD countries", to account for 15 % of all present global residential energy consumption [3]. Moreover, the IEA predicts this total use will triple, to nearly half of all world-wide residential energy use, by 2030.

Unchecked, these accelerating energy burdens for computing and telecommunications systems may place severe stress on economic infrastructures and carry non-neglible environmental impliciations. In particular, emissions from non-renewable fuels originally powering the electrical grid contribute to coupled and quickly worsening crises of natural resource scarcity, agricultural productivity, and public health that accelerate as global temperatures continue to rise [4]. While the computing industry, and in particular the scientific and academic computing community, can make an important contribution to modeling expected impacts and simulate potential solutions to this emerging crisis [5], the projected binge in consumer computing energy threatens to outweigh any potential gains from that effort.

Even more alarmingly, the above estimates of future consumer electronics energy use already miss a critical emerging trend. In particular, a new generation of internet-connected wearables, sensors, robots, and other devices of all varieties, collectively referred to as the Internet of Things (IOT), are poised to come online. Gartner Inc. estimates that another 3.1 billion connected devices will be activated and connected to the data/electrical grid in 2017 alone; this already brings up the total to a staggering estimate, 8.4 billion devices[6]. In addition to raw pressure on the electrical and tele-communication grids, these sort of devices bring fundamental challenges to the future of data transfer, compression/storage, analytics, and integration. According to Gartner:

> "IoT applications will generate extremely high data rates that must be analyzed in real time. Systems creating tens of thousands of events per second are common, and millions of events per second can occur in some telecom and telemetry situations. To address such requirements.... computing platforms.. typically use parallel architectures to process very high-rate data streams to perform tasks such as real-time analytics and pattern identification." [6]

Due to the sheer scale and complexity of this challenge, the emerging IoT computing era will require an entirely new approach to Information Technology (IT) among the world's tech-

nology companies, start-ups, governments, and other institutions. To reduce an exponential burden on centralized systems, any successful new approach "..may require bringing data processing close to the edges ("things") where data is generated... this data of interest may not, as traditionally expected, be kept only at centralized information infrastructures" [7]. In the near term, this transition will probably involve the use of micro-clouds ('cloudlets') as well as distributed or federalized cloud systems, thus reducing reliance on single monolithic data-center; in the medium term, a complete transition towards mobile edge computing (MEC), sometimes also called 'fog computing', is projected especially as 4G/5G global wireless standards come online [8]. In addition to improving efficiency by performing local computations, such distributed and federated systems can also improve fault tolerance by distributing and saving the results of these computations across many nodes in the grid [9].

UNFORTUNATELY, implementing distributed computer systems alone is not sufficient to mitigate these projected future trends. By optimizing only system-level efficiencies, one takes as granted the materials and architecture of modern computing devices when both are fundamentally limited in certain ways. Such is the domination of metal-oxide field-effect transitors (MOSFETs) built from silicon in general, and in particular the modern design of low-power, high speed FET known as the complementary-MOSFET (CMOS), that a compelling case can be made that we presently inhabit the "Silicon Age". Yet, this present dominance hides something of an unfortunate reality about the energy cost of this device. In 1973 Robert H. Dennard postulated that since the overall power density of MOSFET will stay constant during downward scaling, current and voltage will drop as the gate length falls [10]. As these physical parameters have fallen, overall power consumption dropped too, and more transistors are available on a given die (feature size F decreases along with gate length), prices do as well. However, between 2006-7, this trend stopped. Since then, overall CMOS efficiency has continued to slightly increase due to a variety of engineering efforts as high-κ gate dielectrics, strained silicon, and three-dimensional definition (e.g. FinFETs) and stacking. However, with the power density of the newest CMOS now physically *increasing* as F still falls, a major electronics industry shake-up seems to be on the horizon.

Indeed, with transistor feature size now dropping below 10 nanometers, the end to Moore's iconic law- a famous prediction by Gordon Moore that transistor density on an integrated chip (IC) would double every two years- is approaching not as a function of economics but due to physics (thermodynamic constraints). The latest- and last- report of the Semiconductor Industry Assocation, the ITRS, predicts that Moore's law will terminate in 2024 as transistors hit a thermal wall [11]. At smaller than 10nm size, the risk of false bit flips due to high thermal noise becomes increasingly likely, as given by the Johnson–Nyquist formula [12]. However, long before this ultimate physics-wall is hit, a variety of atomic computing schemes that may be fundamentally quantum, and/or analog, in their operation and implementation, may be considered to replace CMOS systems due to superior energy and/or performance. Delaying this changing of the guards are a set of tweaks to physical transistor design collectively referred

to as "more than Moore." Such applications often involve the integration of a variety of exotic new materials into the architecture of the transistor design itself, such as graphene integration to improve speed [13] or carbon nanotube logic devices [14].

At the same time, architectural inefficiencies have multiplied the limitations of this 'Silicon age' computer hardware and software stack. Von Neumann's influential reports on the design for an electronic digital computer [15] emphasized a design in which logic or arithmetic options, collectively known as the central processing unit (CPU) are separated from memory read/write operations both in time and space. The von-Neumann, sometime also called Princeton, architecture dominated relative to more complex proposals, especially since efficient CPU and memory modules could be separately designed and scaled according to different materials and energy requirements. However, as decades have passed and global energy and cooling requirements have become more stringent, this physical separation has become more of an liability than an asset. The fact that all communication between the CPU and memory cells must pass through a shared bus is often referred to as the characteristic *von Neumann bottleneck*, and results in a fundamental speed limit and energy inefficiency for all computers built with this architecture. While CPU cache memory and methods such as branch prediction have somewhat mitigated the time delays due of this limitation, its energetic implications remain mostly unsolved. According to Mark Horowitz, an order of magnitude more energy is spent transferring/accessing the data in memory or obtained in the CPU, then in the raw energy cost of the fundamental operations needed to obtain them (e.g., write operation in memory, or the cost of powering transistors performing arithmetic operations) [16]. Even worse, the von-Neumann paradigm has also created inefficiencies at the level of the hardware level languages, and later software systems, used to operate upon this foundation. In his ACM Turing Prize lecture, John Backus described this as follows:

> "The von Neumann bottleneck.. is [not only] a literal bottleneck for ... data traffic .. but, more importantly, it is an intellectual bottleneck that has kept us tied to word-at-a-time thinking instead of encouraging us to think in terms of the larger conceptual units of the task at hand. [P]rogramming is basically planning and detailing the enormous traffic of words through the von Neumann bottleneck, and much of that traffic concerns not significant data itself, but where to find it."[17].

MOVING beyond these bottlenecks requires a fundamental re-thinking of the way(s) we compute and the substrate(s) we use to do so- a difficult task. Yet, if navigated successfully, visionary new solutions might simultaneously solve the linked conundrums of rising energy budgets, over-reliance on centralized systems, and over-reliance on outdated architectures. While individual logic and memory operations with nanodevices have been implemented for at least a decade, in particular using carbon nano-tubes for random access memory and semi-conducting nanowires as configurable field-effect transistors [18, 19], these designs still separate memory and computing functions. On the other hand, emerging designs broadly

referred to as in-memory computing which directly co-integrate computing and memory functions, have only so far been considered in the context of 'in database' computing operations with dynamic-Random Access Memory (DRAM) devices [20]. The development of a new generation of nano-architectures that attempt similarly hybrid operations within the context of the physics or dynamics of these devices themselves remains a critical and unfinished research task. In particular, emerging types of memory devices known as resistive switching devices open the possibility of merging existing 'hierarchies' of memory access into a unified template for on-chip computing and data storage, which could bring massive gains in computing performance, energy efficiency, and density [21]. Critical advantages of these nanodevices for building new types of architectures are their intrinsic non-linearity (we can build and chain threshold operations with them), their non-volatility which can yield extreme energy efficiency, and the ability to perform biologically plausible operations such as hebbian learning in-situ (within the device), rather than emulating these behaviors in accompanying silicon hardware [22]. Ultimately, these new types of architectures will be critical to realizing an Internet of NanoThings (IoNT), in which nanoscale memory-computers and nanoscale sensors localize energy production and use, computation tasks, and distributed storage [23]. Not only would such an IT strategy avoid the negative impacts from existing or linear scaling with CMOS systems, it could open the doorway to revolutionary applications in healthcare, environmental and agricultural monitoring or repair, and economic optimization.

THIS thesis explores the use of a new class of non-volatile memory (NVM) computing elements, also referred to as memristive nanodevices or memristors, as an attractive building block for IoNT architectures. Arrays built with these memristive or NVM devices are intrinsically attractive from the energy-savings perspective, as they require zero power in standby (non-volatile) mode, unlike modern staged memory-CPU systems that require power to preserve states and to power-up/power-down memory cells. In addition, well-engineered emerging NVM devices are fast, extremely durable, and scalable down to ultrahigh density. For these reasons, NVM devices integrated in dense , three-dimensional arrays are actually in the process of industrial realization to replace Flash memory in laptops and phones. Some of the claimed benefits of pre-industrial NVMM arrays include 1000x faster writes, 20x less power consumption and half the total die size of earlier storage systems [24]. An industrially available NVM system, Intel's 3D XPOINT, has already been integrated in a solid state drive (SSD) product line called Optane [25].

Fig. 1 (a) shows an NVM three-dimensional memory array where the yellow contact lines are known as bit and word lines and the selection of particular indices on the array allows for devices/cells to be written (green), deleted (red), or read out. This is already exciting, yet in the following chapters I argue that these devices represent a serious opportunity for building neuromorphic architectures and may be under-used in the context of stateful storage or random access memory (RAM) architectures. In fact, memristive nanodevices have special properties that make them an attractive hardware implementation reminiscent of biological synapses;

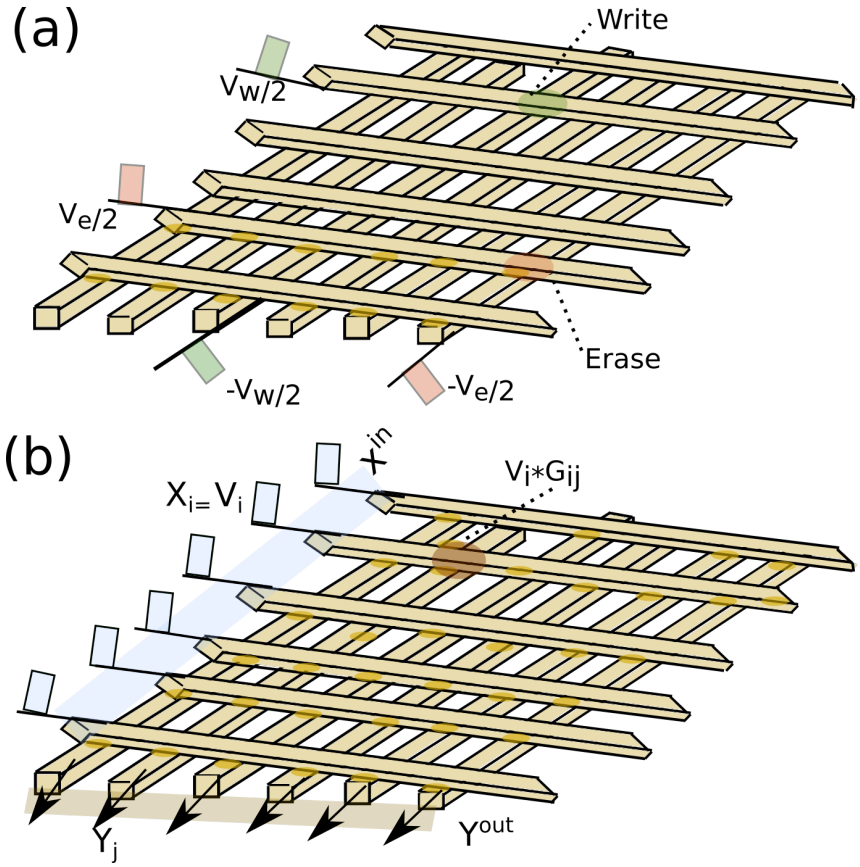


Figure 1: (a) Depicts a crossbar where word line (WL) and bit line (BL) are specified to either write (green) or erase (red) a stored cell of information, in this case a single stored bit. (b) Depicts a crossbar in a neuromorphic context, where a set of input voltages $\boldsymbol{x}$ provides an output ($\boldsymbol{y}$) which is functionally a vector-matrix product. As highlighted, each cross point implements an elementary operation where the conductance G is multiplied by the pre-synaptic voltage V_i to contribute to this product collected at the post-synaptic (bottom electrode) wires.

like real synapses they learn/remember meta-stable ionic concentrations, and possess intrinsic non-linear dynamics [26, 27]. Meanwhile, while ReRAM is fast and energy efficient, it still does not perform its own learning or inference. In the following chapters I demonstrate many examples of architectures that do just this, and in fact have been designed expressly to perform their own learning locally. As already discussed, such a non-von-Neumann approach could massively reduce the transfer of data between physically separate memory and computing cells, further saving energy. In Fig. 1 (b), an essential non-von-Neumann operation used in all of these architectures is noted. The operation, which has been variously referred to as a forward-inference read, or an on-chip matrix-vector operation, uses a vector of simultaneously presented voltages $\boldsymbol{x}$ to almost instaneously read-out a set of outputs $\boldsymbol{y}$. The mutliply operation is electrically performed at every crosspoint by Ohm's law, while the accumulate is electrically performed at each post-synaptic (output) line by Kirchoff's law [28]. As this operation is a crit-

ical feature of many artificial neural network (ANN) models, the ability to perform it in-situ in hardware is an unprecedented advantage.

Preview of key topics explored

In the following chapters, the vision of on-chip (local) learning with memristive devices is built up gradually. Chapter 1 provides a strong conceptual foundation by introducing the basic concepts of neuromorphic engineering, describing material aspects of various memristive devices, describing a variety of conceptual models of computation (artificial neural network models), and lastly illustrating a general survey of methods for on-chip learning. Chapter 2 focuses in depth upon the computing possibilities of one particular device, a powerful organic thin-film memristive device, showing its concrete integration in both simulated and experimental local learning schemes where it always plays the role of nanosynapse. Chapter 3 zooms out to larger neurmorphic artificial neural network (ANN) architectures- all simulated- which would allow these basic principles to be scaled to difficult tasks encountered in the world of machine learning. In this chapter, a particular focus is put on the non-ideal aspects of device behavior that might constrain these systems at scale. Finally, Chapter 4 proposes new varieties of architectures that use timing dynamics, at either the device or system level, to reach new levels of learning performance and/or energy-savings.

Summary of key contributions

- A novel organic memristive nanodevice has been characterized and modeled, and a suite of simulations were performed to understand more about its complex behaviors. These simulations were overall promising, and thus encouraged us to go ahead with an experimental demonstration. In the process, we formalized a compact model.
- We experimentally implemented an elementary learning system with several of these organic memristor devices. The learning system was built to learn linearly separable functions automatically. We analysed in great depth the imperfections of the devices in this learning set-ups and how these imperfections affected learning outcomes. Notably, we developed different learning rules to account for a key physical contraint when learning (device asymmetry).
- Building upon our simulated and experimentally realized nanodevice learning primitive, we expanded our analysis through space and simulated complex multi-layer memristive learning systems across a wide variety of key parameters. Ultimately, we have compared and contrasted conventional on-chip multi-layer perceptron architectures with random first layer, trained second layer systems (often called NoProp or ELM in the literature), revealing intriguing trade-offs between accuracy, size, energy, and overhead requirements.

- Due to the promising performance of the nanodevice-built projection-regression systems, we also looked for additional ways to enhance this system. First, on the device level, we looked at the potential for the first-layer weights to be imprinted by exploiting a unique plasticity transition made possible by a silver ionic electrochemical metallic cell. Indeed, this system seems to outperform the fully random system.

- Second and finally, on the systems level, we evaluated the impact of increased neuron complexity and sparsity (less than full synapse connectivity) to reduce overhead. Implementing this design allows us to achieve a 28x reduction in the crossbar size required to implement on-chip classification at high accuracy.

Chapter 1

Essential Neuromorphic Concepts, Devices, and Methods

"Either mathematics is too big for the human mind, or the human mind is more than a machine."

Kurt Godel

"EXTENDING *upon the global context of the introduction, this chapter serves as an overview of the rapidly emerging field of neuromorphic computing field: its objectives, its models of computation, and their possible physical realization(s). Taken together, the confluence of neuromorphic design and emerging concepts from material physics emphasizes the importance of alignnment between physical device behavior, scaling dimensions and trends, and abstract models of computation. In particular, the broad class of devices considered in the context of this thesis, memristive devices, are introduced in the context of this grand challenge and their favorable aspects highlighted.* "

THIS CHAPTER presents a conceptual background in the field of neuromorphic engineering (1.1), and furthermore, provides an introduction to the concrete class of nanodevices (and their associated phenomena) used throughout this thesis (1.2). Lastly, a variety of non-VonNeumann computational primitives and concepts are introduced so as to provide adequate conceptual background on how such computing systems work (1.3). This chapter does not serve as a comprehensive review to the theory or practice of neuro-computation, and contains blind-spots. Pedagogically, it has instead been oriented towards describing in advance the essential concepts which shall be important to understand the later chapter's works.

1.1 The Neuromorphic Imperative

1.1.1 Carver Mead's Vision

As Carver Mead noted in the 1980s, the cost of a computer's single operation was then $10^{-7}J$, yet the entire computer's operation per cycle is $10^{-5}J$, or two orders of magnitude more expensive (due to scaling, these numbers would be slightly lower today) [29]. Mead's vision for the future of electronics rests upon the assumption that, even assuming ultimate scaling (100 fold higher density), surpassing $10^{-9}J$ per elementary on-chip operation, and $10^{-7}J$ system wide, our electronic systems would remain in the best case around *10 million times* less efficient than the human brain. With 10^{16} synapses, about as many complex operations per second, and a total energy budget of a few Watts, (thus $10^{-16}J$ per operation), the brain remains an un-reached Pareto frontier compared to all of our current schemes. Mead's "unavoidable conclusion" is that we have something "fundamental to learn from the brain about a new and much more effective form of computation" [29]. In the 25 years since this vision was articulated, a good amount of progress has been made towards realizing it, primarily in the context of silicon neuromorphic designs, as the next section addresses.

1.1.2 Analog Neuromorphic Implementations in Silicon

Some of the first implementations of Mead's visions utilized standard MOSFET technology in the sub-threshold regime. Using these principles and other standard electronic components such as capacitors, simple spiking neural models for integration in larger systems, e.g. the leaky integrate and fire (LIF) model, can be constructed [30]. Moreover, since accessing transistors in the sub-threshold requires smaller currents than in standard digital arithmetic logic unit (ALU) architectures, very large scale integrated systems (VLSI) built from these building blocks can be extremely energy efficient [31, 32]. While simple pattern recognition has been achieved using spike-based learning rules [33], achieving industrial results requires general purposes architecture, e.g. multi-core designs that separate spiking neurons and synaptic stored weights. A few examples of these sorts of architectures recently proposed and in various stages of de-

velopment include NeuroGrid at Stanford [34], and the ROLLS spiking neural system developed at the University of Zurich [35]. In addition original learning schemes such as recurrent (attractor-based) schemes, have also been introduced and evaluated in the context of analog CMOS neuromorphic systems [36].

1.1.3 Digital Neuromorphic Implementations in Silicon

Analog VLSI implementations exploiting transistor physics provide one possible path to reaching the neuromorphic vision, but recent efforts have also focused on entirely digital implementations. Presently available, general purpose systems that can realize such designs include Field-programmable gate arrays (FPGA), which incorporate fast, on-site memory for weight storage, usually with Flash or electrically erasable non-volatile memory (EEPROM) devices. In addition, special full-purpose CMOS neuromorphic architectures including SpiNNAker at the University of Manchester [37], and IBM's TrueNorth [38] have been desinged to meet key neuromorphic criteria including parallelism and energy efficiency. On the latter platform, a whole host of non-vonNeumann neuro-inspired algorithms have been successfully ported, including hidden Markov Models, restricted Boltzmann Machines, and multi-layer perceptrons [39].

1.1.4 Moving beyond Silicon in Neuromorphic Design

All of these systems rely upon well developed commercial device technologies- CMOS, DRAM, SRAM, and EEPROM- to realize reconfigurability on many scales, to compute inter-neuronal weights (synapses) and to simulate intra-neuronal dynamics (activation functions) [40]. However, in the last decade, new nanofabrics incorporating emerging memory and logic devices at multiple scales have helped open the doorway to post-silicon, or hybrid silicon-nanodevice, implementations of the neuromorphic vision. In terms of future reconfigurable computing systems, such designs may improve the speed and efficiency of existing FPGA implementations, open the way towards a new type of field-programmable analog array (FPAA) design built with in part with nanodevices, or even lead to bespoke new analog computers built entirely with nanodevices. In any of these systems, critical design assets of emerging non-volatile memory (memristive) devices include the possibility for extremely dense information integration, strong energy efficiency due their non-volatile operating modes, and flexible operation between analog or digital modes, as suggested in [41, 42]. The next section provides far more evidence of the promise of these devices from the material physics perspective.

1.2 Memristive Devices: A Neuromorphic Building Block

1.2.1 Early Interest in Two-Terminal Nanodevices

Long before nanodevices were shown to exhibit memristive properties, there was already intense interest upon the use of two-terminal, non-volatile nanodevices in the context of emerging molecular computing proposals. In particular, much of the early research concerned semiconducting nanowires or carbon-nanotubes [19], but these were usually meant either to imitate transistor designs or act as solid-state memory. An exception was the early neuromorphic considerations of Likharev [43]. In 2003, nanoscale molecular switches were experimentally realized in a grid or crossbar configuration, with top and bottom wires sandwiching a rotaxane active switching layer [44], making such speculations far more feasible. Then, in 2005 Strukov and Likharev, proposed an ambitious design called CMOL, which proposed that this variety of molecular two-terminal devices could be directly paired with CMOS logic circuits into reconfigurable architectures inspired by field programmable gate arrays (FGPAs) previously built with CMOS [45]. This design was further expanded in (CrossNets), which estimated top-performance and energy draw of these systems, and proposed a variety of concrete neuro-inspired tasks and architectures which could be attempted [46].

1.2.2 The Theoretical Memristor

In 1971, Leon Chua predicted a future fundamental circuit element [47]. Of the four basic electronic state variables- current (i), voltage (v), charge (q), and flux (φ), he asked why there were mathematical relationships between most (yielding resistor, inductor, and capacitor), and yet a relationship between q and φ did not exist. In turn he introduced both the characteristic φq curve, and its device symbol. At a given time t, the voltage drop of a charge-controlled memristor is given by

$$v(t) = M(q(t))\mathrm{i}(t) \tag{1.1}$$

$$M(q) = \frac{d\varphi(q)}{dq} \tag{1.2}$$

Current through a given memristor device and the evolution of the internal state variable W are given by:

$$i(t) = W(\varphi(t))\mathrm{v}(t) \tag{1.3}$$

$$W(\varphi) = \frac{dq(\varphi)}{d\varphi} \tag{1.4}$$

Thus, at t, the state variable depends on the total integral of all previous changes in, respectively, charge or voltage. This implies the device has an intelligent, or stateful memory of what has occurred before [48].

In 2008, a hugely impactful Nature article, summarizing the work of researchers at Hewlett-Packard (HP) Labs emphasized that the missing hypothetical (theoretical) memristor had been experimentally realized in the form of titanium dioxide switching cells [49]. Since then, criticisms have been levied against the claim that this experimental device accurately corresponds

to the full theoretical circuit model, and whether the ideal theory comports to known facts of condensed matter physics [50]. Of particular relevance to experimental designers is that the original model proposes a completely smooth pinched-hysteresis loops, while real memristive nanodevices usually have a non-continuous hysteresis curve bifurcated by thresholds. In [51] this issue was further clarified by Chua in differentiating between the circuit theory of ideal non-linear circuit elements, and real memristive devices, non-volatile analog memory devices which implement Ohm's law in a state-dependent way. The 'real' memristor model is:

$$v = R(x,i)\,i \tag{1.5}$$

$$\frac{dx}{dt} = \boldsymbol{f(x,i)} \tag{1.6}$$

Where $\boldsymbol{x}$ is a set of internal state variables independent on current/voltage.

In fact, this simplified formluation is very similar to Pershin and DiVentra's formulation of memristive, memcapactive, and mem-inductive systems from condensed matter theory (Kubo response theory) [52]. In its most general and well-recognized form, all mem-elements might be expressed as:

$$y(t) = g(x,u,t)\,u(t) \tag{1.7}$$

$$\dot{x} = f(x,u,t) \tag{1.8}$$

where f is a vector of internal states corresponding to the relevant behavior (ion diffusion, etc), $u(t)$ is the input, and $y(t)$ is the output as a function of time [53]. For memristive devices, then, which have meta-stable internal states relevant to their electrical activity expressed as conductance, $u(t)$ is current, $y(t)$ is voltage, and $\dot{x}$ would be a momentary conductance value. In fact, f can be further understood in the case of memristive devices by clarifying a unitary vector $\boldsymbol{x}$ of all relevant state variables. Given $\boldsymbol{R}$, the vector of all atomic positions of relevant ions, an applied electrical field $\boldsymbol{E}$, and assuming a Newtonian treatment of force on each ion, $\boldsymbol{x_1} = \boldsymbol{R}$, $\boldsymbol{x_2} = \frac{d\boldsymbol{R}}{dt}$, then

$$I(t) = G(\boldsymbol{x_1}, \boldsymbol{x_2}, t)(V(t)) \tag{1.9}$$

$$\dot{x_2} = f(\boldsymbol{x_1}, \boldsymbol{x_2}) \tag{1.10}$$

When we simply define $\boldsymbol{x} = \boldsymbol{x_1}, \boldsymbol{x_2}$, the easily recognized form is again obtained:

$$I(t) = G(\boldsymbol{x}, t)V(t). \tag{1.11}$$

Other electrical 'mem-elements' , e.g. memcapacitors (capacitors with programmable/meta-stable states) can be defined according to the appropriate input-output pairs. In fact, symmetric derivations were also made on the basis of relevant internal state variable equations for mem-capacitance devices (storing energy or charge) or mem-inductive devices (storing flux linkage). For the purposes of this work, we emphasize that we do not depend too strongly on any one theoretical construct; however, our studied devices, as physical nano-conductive/nano-

ionic map most closely to Pershin and DiVentra's model.

1.2.3 Physically-Realized Memristive Nanodevices

As noted in [54], a variety of physical memristive device 'families' exist, each of which depend on a variety of internal processes of ionic drift/diffusion, magnetic response, crystallization, etc. Care must be taken in this case to emphasize the true physical cause of switching phenomenon, and not electrical effects due to the electrode, or corresponding activity such as nano-scale Joule heating. Following this philosophy, we explore the physical mechanisms, advantages and disadvantages of a few of these families. Particular attention is paid to the first of these families, redox-based or filamentary memristive devices, since the two particular nanodevices studied mostly closely in this work fall are members.

1.2.3.1 Redox-based resistive Memories

Redox-based resistive random access memory, or ReRAM, is an emerging class of nanoelectronics elements which tend to rely upon filamentary behavior in order to realize multi-state switching. Waser places ReRAM devices in the context of the broader class of MIM memory devices, in which M are good electron conductors, often standard metallic electrodes, and I is an insulating layer and a medium which allows for ionic conduction and/or growth [55]. MIM memory systems are known to possess hysteretic switching and are broadly divided into bipolar and unipolar classes; while the former depends on the polarity of the applied voltage or current to switch, the latter does not (is symmetric). A wide variety of redox switching scenarios exist; in a typical such system, an active ionic electrode (anode) is reduced and forms filaments as it moves through the insulating layer towards the inert cathode. This particular variety of cell is called an electrochemical metallization cell (ECM) [56], which have also been referred to as Conducting-bridge memories (CB-RAM) in the literature as well. In ECM cells, the exact mechanism of dendritic filament formation and growth depends greatly on the ionic/insulating layer (e.g. oxide, sulfide, or selenide), leading to a variety of device properties [57]. Another variety of ReRAM exists, which relies upon the movement of migrating anions (negative ions) rather than cations; this typically happens in transition metal oxides. This is the case for HP's iconic TiO_2 device in which oxygen vacancies migrate between the two electrodes, and related oxide ReRAM computing devices built from Vanadium and Tantalum oxides [55].

According to Waser [58], all the above device varieties have the following important properties and advantages:

1. Multi-level switching. Because these devices rely upon motion of individual ions and demonstrate meta-stable states as filaments are formed and destroyed, they can intrinsically be written (and read) from many points. In principle, the ultimate limit would be Landauer conductance steps, which some studies on atomic switches (ECM) approach.

2. Fast write speed. Because again single atoms are being moved, even extremely small voltage pulse (widths) have a discernible effect on conductance. ReRAM switching times can be on the order of nanoseconds, and are usually less than $100ns$.

3. Excellent scalability (bottom feature size). Since the ultimate computing frontier is atomic and again ReRAMs are intrinsically exploiting nano-ionic dynamics, all well-scaled ReRAM devices should in principle be $F < 10nm$ (less than $10nmx10nm$).

In short, ReRAM devices work because of their dynamic filamentary behavior, based on the movements of nano-ions, and this gives them the ability to manifest rich meta-stable states; from this, we can build powerful computers.

ReRAM devices also have signature disadvantages. One issue is that due to the wide variety of possible materials and the electronic potentials needed to move ions, low-voltage and low-current operation is not always possible, which means energy-dissipation per write in these systems may be higher than competing approaches. Second, ReRAMs are arguably the most intrinsically variable and prone to retention issues (e.g. conductance drift) of all emerging memory types. Since the devices rely upon atomic mechanisms for their basic operation, inter-device variability is effectively unavoidable, as a wide variety of physical parameters yield different operating conditions for filamentary formation/destruction within every single ReRAM device. When scaled to large arrays, variation in key parameters, e.g. G_{ON}, the maximum typical conductance (corresponding to a complete filament), and G_{OFF}, the minimum typical conductance (corresponding to no filament) are usually significant [59].

Recent studies provide new insight on the formation and growth of filaments, which could be critical to optimizing ReRAM computing platforms. In [60], transmission electron microscopy revealed two different varieties of filamentary growth, and suggested that since the narrowest filament growth was near the dielectric (inert electrode), this area may need to be further optimized in future devices. A three dimensional study of conducting-bridge filament formation was made using conducting atomic force microscopy (c-AFM) which suggested that cation-transport was the rate-limiting aspect of their studied device, and suggested that optimizing cation mobility could lead to faster and more efficient ReRAM devices [61]. A similar study using c-AFM to examine filamentary formation in a sub-set of ReRAMs called valence change memories (VCM) found that multiple filaments were often growing and competing simultaneously [62]. Drawing on some of these insights, recently various improvements to traditional ReRAMs including a custom Schottky barrier at electrodes and specific doping schemes to improve filamentary mobility have been proposed [63].

1.2.3.2 Phase Change Memories

Phase-change materials (PCM) memories, sometimes also referred to as chalcogenide memories or ovonic memories in the literature, rely upon phase transitions within a small glass 'plug' which sits in between two electrodes and possesses two structurally distinct material states, usually, an amorphous and a crystalline one; these states of matter possess dramatically different electronic properties and are switched using heat [64]. The implementation of such materials into memories depends upon developing a fine control of tuning between the energetically stable crystalline states and meta-stable amorphous states using the interplay of heating and cooling phases. The material most widely used in PCMs is germanium-antimony-tellurium or GST, which sits in an insulating middle layer and is connected between an upper and lower electrode. In this device and many other similar ones, the low-resistance state occurs when the device is an a crystalline state (current flows, '1'), and the high-resistance state occurs when the device is in the amorphous (less or no current flows, '0'). In order to program the device (SET, $0 \rightarrow 1$), a particular voltage determined by Joule heating sets the device to low-resistance at a temperature above the crystalline phase, but below melting point, so it cools back into the crystalline phase. To erase it (RESET, $1 \rightarrow 0$) an even higher voltage (above the melting point) is used to bring the device into the amorphous state. These operations are almost instantaneous, with the only major delay being the 'cooling' wait; however, this time also reduces as devices are scaled.

As a result, well-engineered and scaled phase-memories have several favorable physical properties including extremely fast (<10 ns) switching time and long stability (5-10 years) [65, 66]. Based on these favorable properties, they are being considered as a replacement to traditional floating-gate flash memories for conventional memory storage [64]. In addition to binary memory storage, can also be used in an analog fashion by applying extremely small programming pulses to access a range of intermediate states [67, 68]. PCM devices are presently being used in both binary and analog-mode for neuromorphic purposes, as discussed in 1.4.

1.2.3.3 Spintronic or ferroelectric memories

The most well-known and industrially ready magnetic memories are spin-transfer torque RAM (STT-RAM), multi-layer magnetic tunnel junction devices in which a 'free' magnetic layer can switch between one of two orientations (making it a binary device). The device is current-controlled, very fast, and has great endurance. Since it can also be scaled to very small feature size [69] and is compatible in ultra-dense cross-point architectures [70], this technology is currently another candidate for replacing conventional (e.g., Flash or DRAM) memory storage. In addition to conventional memory applications, spin-based devices are being considered to build next-generation logic circuits [71, 72] and in the neural network context, to use spin devices either as synapse [73] or as neuron [74].

Another interesting magnetic synapse candidate are superparamagnetic tunnel junctions,

atomic scale in-plane magnetized junctions that exploit the switching behavior of a free layer ($CoFeb$) pinned in between magnetic barrier layers (MgO). These devices can be switched with very low currents, making them leading candidates for future computing systems. Of particular note is that oscillator synchronization [75], and phase-locking [76], may be useful to build ultra-low power circuits that utilize stochastic effects intelligently.

Lastly, a very recent and promising type of nanodevice is the ferroelectric memristive device realized by Julie Grollier and her team [77]. In contrast to the STT-RAM device above, this electronic device can be addressed at multiple analog levels. In addition to being an attractive solid state memory device, these sort of spintronic devices may make ideal candidates for integration into a variety of emerging neuro-inspired architectures [78].

1.2.3.4 Carbon Nanotube Memories

CNT-based non-volatile memory storage were proposed as early as [19], and their memristive behavior (properties) was subsequently demonstrated [79]. Their integration in a neuro-inspired circuit was first proposed in [80]. A variant of this design was since experimentally realized [81]. Advantages of these devices include high speed due to electrical mobility, while a strong disadvantage is the lack of easy tunability or controllability of intermediate conductive states. Often these require exotic physical mechanisms to reduce conductance or burn the devices, such as using lasers [81].

1.2.3.5 Nanoparticle Assemblies with Memristive behavior

Soon after the original solid state memristor device demonstrations, typical memristive hysteresis curves were also observed in nanoparticle electronic devices. In [82], the authors detected this behavior in collections of crystalline magnetite nanoparticles and notably hypothesized the memristive effect was due to the different mobilities of two ionic species moving throughout the crystalline lattice. In [83], another more complex nanoparticle system with memristive effects was realized, this time with gold nanoparticles suspended in a thin-film of organic material called pentacene. The device, referred to as a nanoparticle organic memory field effect transistor (NOMFET), displays meta-stable electronic states consistent with the formation of many filaments through the active thin film layer.

1.3 Neuro-inspired architectures and algorithms

1.3.1 Connectionism's past and future

One might reasonably argue that McCulloch and Pitt's iconic paper [84] gave birth to the field of neuro-computing. In a somewhat radical suggestion, the piece suggested that, not only was the brain comprehensible, but that any conceivable (neural) network may be "treated by means

of propositional logic." Even more astoundingly, the paper already made a serious distinction between such networks with and without circles (recurrency), and predicted that, by using recursion, such circuits may make networks complete. From the broader perspective, McCulloch and Pitts' work was also the birth of a powerful variety of computational thinking that was and is referred to as *connectionism*. This approach to cognitive computation emphasizes the critical role of networks in determining the algorithms of learning. As a core principle it emphasizes that macro-states, far from being 'classical' phenomena, are really caused by underlying distributions of many micro-states or features. While this suggestion was at first considered as reductionist and heretical, the past half century's advances have treated it kindly [85].

From a more abstract computational perspective, formal investigations of the space of computable numbers, functions, and vector spaces remained an open and thorny issue. In 1937, Turing noted that the computation of every logical gate (AND, NOT, NAND, XOR..) can be embodied in the activity of a Turing Machine (TM), a theoretical construct composed of a readable/writable set of states (alphabet), tape head which reads and writes these states, and a table of transitions (more colloquially, a computer program) [86]. By combining the correct number or type of gates into the operation of this program (for instance, NAND gates alone are sufficient), a universal computer is born. In addition, a universal Turning Machine (UTM) can be constructed by chaining the operation of many smaller TMs. As a corollary, the Church-Turing Thesis argues that anything that is computable, can be represented by an equivalent TM or UTM design. However, Turing models may be an inefficient way to model many physical systems due to exponential complexity issues. In fact, physical computing in biological organisms is a sequential process that occurs through time, with the substrate itself evolving and adapting. Although little known, Turing himself was rather sympathetic to some of these connectionist arguments, and fascinated by a variety of biological computing and structuring processes such as morphogenesis [87]. In the last decade, renewed theoretical interest in the power of bio-computation has led to a variety of amendments to the 'strong' Church-Turing thesis [88, 89].

Broadly speaking, *unconventional* computing considers how various physical phenomenon or systems, such as chemical/biological networks or quantum systems, may enhance, expand, or directly challenge the performance of classical (digital, turing-inspired) computers [90]. *Neurormorphic* computing is one sub-set of unconventional computing which considers how the brain's massive parallelism, spatio-temporal processing modes, heterogeneity of sub-systems, and hierarchy may inspire a new generation of molecular computers inspired by the brain's chemical mechanisms of computation. The next section sketches a general outline of the biological computation through a family of conceptual models of learning collectively known as artificial neural networks (ANNs). In a sense, these conceptual models have become powerful experiments in neuro-computation themselves.

1.3.2 Artificial Neural Networks

ANN or ANN-like structures are topologies (structures) that possess dynamics (behavior) and are evolved according to certain rules. The two essential features of any artificial neural network are its nodes or information processing units ('neurons'), and its adaptive connections (synapses). Next, we expand upon this idea in biological terms.

1.3.2.1 Biological inspiration

A schematic of multiply connecting biological neurons and synapses in visible in Figure 1.1, and related context provided in Table 1.1. The following concepts, inspired by neuroscience research on how real neural networks work, are critical to understanding how artificial neural networks function as well:

- **Topology**: A community or population of neurons are connected together in a particular topology; of particular relevance is the order of information propagation, e.g. in Fig. 1.1 neurons (a)-(d) pass along information to integrating and receiving neuron (e). This directionality of neurons is directly relevant to adaptation or learning, as explained in Section 1.3.3. Note, however, that neurons can also create backwards information flows; for instance, a downstream neuron may connect to a third neuron which connects back around to the original neuron- creating recurrence.

- **Integration**: Based on the spike activity of preceding neurons and the connectivity matrix/strength of various synapses in the pre-soma dendritic arbor, each neural cell body (soma), (a)-(e) in Fig. 1.1 may fire at a given time, if and only if the total excitation on its membrane exceeds some critical threshold. The biological integration and firing procedure is complex, and relates to the spatio-temporal integration of excitatory post-synaptic potentials (EPSP) along the membrane due to the activity of various ionic channels.

- **Information Propagation**: Neurons communicate information through these critical spikes or action potential events, as visible in the box (iii) and noted emanating from each of the active neurons in Fig 1.1 (b,c,e). Spikes, once generated by the cell body, travel forward along the axonic system towards other neurons with which it is interconncted with via axonic synapses.

- **Learning**: Learning in a neural network takes place critically on the level of synapses (it may happen at other time-scales and locations too). Synapses are the junctures between neurons, as pictured in the red and green dots in Fig 1.1 box (vi). They change their strengths relative to the activity (spiking patterns) of neurons, which physically occurs through the activity of chemical communicators known as neurotransmitters. Synapses adapt and integrate information differently depending on their place in the neural net-

work. For instance, with regards to the synapses pictured in the box (vi), (e) is post-synaptic, and (a)-(d) are pre-synaptic. Thus, synapses embody bi-directional information about system connectivity. In addition, broadly there are two classes of synapses: dendritic synapses which are receiving information in arbor structures (i), and axonic synapses which are delivering it in the terminals of the neuron (v).

Artificial neural networks are in general rough approximates of the complexity of real (biological) neural networks, and in particular two short-comings of traditional ANN models are worth highlighting. First, while in biology the structure of neural networks is intrinsically alterable and new wiring patterns (neurogenesis) is known to be a key component of memory formation and consolidation [91], the vast majority of ANNs have static topologies (e.g., new connections may not be allowed to form, or the basic topology itself may be set). However, some varieties of ANNs rely upon evolving topologies [92], and a recent proposal considers the evolution of spiking neuronal systems [93]. Second, while according the original McCulloch-Pitts model the dendritic branch is treated in most artificial neuronal systems as a simple weighted sum, this particular sub-system has extremely complex dynamics, and seems capable of performing complex temporal mapping operations in its own right [94–96].

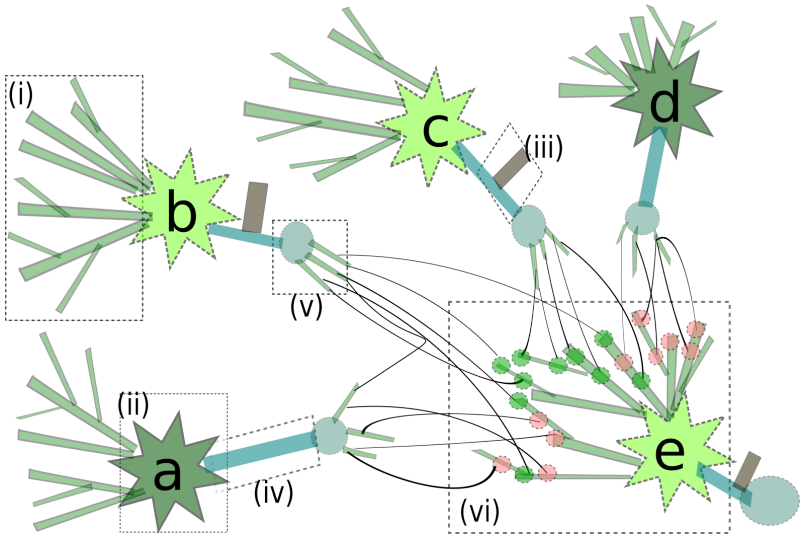


Figure 1.1: This illustration suggests some of the very basic aspects of neuronal structure and function that are crucial to an understanding of ANN systems. Here, neurons are typically composed of 4 main sub-systems: a dendritic arbor which pre-processes (i), the central neuron cell body or soma (ii) which accumulates inputs and in response may output a characteristic spike response (iii), which is forwarded-propagated along the axonic system (iv), until it ultimately reaches the axonic terminal system and forward communicated to other arbors (v). (vi) shows a field of synapses in the process of adaptation; more detail is given in the text.

1.3.2.2 ANNs: Hypotheses of Adaptation

Any intelligent system or organism has the underlying task of predicting or inferring future events based on past ones; this is critical to allow for survival and/or reproductive success, in the case of living organisms, and thus evolution has optimized for it. Succeeding at this meta-task requires an adaptive system/organism to possess a native understanding or model of the true causes of past observations/data. In the case of neural networks, the embodiment of this model and the algorithm that is used to achieve it an active topic of research. For instance, while one model of the brain's canonical learning circuits has been hypothesized [97], it is not universal. A central puzzle concerns how the brain manages to efficiently encode, transmit and preserve information on different temporal and spatial scales [98]. While a variety of theories including neuronal orientation selectivity or 'tuning curves' [99, 100], individual spike encoding, and spike-rate based encoding [101] have been used to predict the behavior of individual neural circuits, an active debate still persists about which is most explanatory [102].

With the atoms of neural computation mostly unenumerated, there is a great deal of interest in expanding our patchwork understanding of the brain's central algorithms or learning methods [103]. Nonetheless, there is widespread appreciation that the fundamental engine of this algorithm or ensemble of algorithms must be continuous adaptation at many temporal and spatial scales. Adaptive or learning methods may be broadly divided into cases. The first case, supervised learning, provides neurons with a directly available teacher signal which communicates a target behavior/output; the difference between this ideal output and the actual provides the engine of learning/adaptation. This signal can either be externally provided, for instance by a teacher brain circuit/region, or automatically generated by the environment in the form of positive or negative feedback communicated back to the brain circuit. Simple reinforcement learning, e.g. trial and error based learning, has indeed been observed in the adaptations of synapses in the oculomotor system [104].

In the second case, either no teacher signal is ever seen, or it is rarely seen; instead, local dynamics and information steer neurons to adapt. The two cases, unsupervised and semi-supervised learning respectively, typically use biologically-inspired adaptation methods (introduced rigorously in following section), or statistically-derived inference models- e.g. learning with Bayesian or probabilistic graphical models [105, 106]. Moreover, recent research suggests that the wiring patterns of neural circuits and the natural properties of stochastic phenomenon within them may be exploited to naturally perform probabilistic inference [107].

Given this high-level context, the following sections concretely introduce the abstract computational models (ANN types) referred to in the rest of the thesis. We start with the two most classical adaptation rules, Hebbian learning and the ancestors of gradient descent, which are the engines for most un-supervised and supervised learning methods, respectively. Following introduction of these classical methods/structures, progressively more complex modern-day networks are introduced, finally leading to a discussion of deep neural networks, recurrent networks, and reservoir computers.

Table 1.1: Key components of artificial neural networks

Component	Key Function(s)
Neuron	Information Integration
Cell Body/Soma	Activation Function / Threshold
Pre-Neuron Spike Train(s)	Input; Information channel(s) to be integrated
Post-Neuron Spike Train	Output; Information channel(s) to be communicated
Dendritic Synapses	Pre-processing and weighting of incoming connections
Axonic Synapses	Post-processing and weighting of outgoing connections

1.3.3 Hebbian Learning

The classical rule of actual neuronal spike-learning, the Hebbian rule of learning, has been stated colloquially as follows: 'neurons that fire together, wire together.' More formally, Hebb stated that a cell A is near enough to excite a cell B repeatedly, such that "some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" [108]. This positive association or correlation is known as long-term potentiation or LTP, and its inverse (anti-correlation) as long-term depression or LTD. Mathematically, the classical Hebbian principle is extremely simple:

$$\Delta w_{i,j} = \eta x_i(t) x_j(t) \tag{1.12}$$

Where $x_i(t)$ is the activity of the pre-synaptic neuron at some time, $x_j(t)$ is the activity of the post-synaptic neuron at the same moment, and η is a scaling parameter. According to this formulation, if neuron i and j are simultaneously firing they strengthen together (associate); if one is firing and the other not they lose connection strength (dissociate); else, there is no change. This basic dynamic is demonstrated in Fig. 1.1, box (vi) where the (green) synapses strengthen and are connected between the co-active synapse pairs (e) and (c) and (e) and (b), and the rest (red) connected to the inactive pre-synaptic neurons weaken.

A further generalization of Hebbian learning explicitly integrates a teacher signal. As given in [109], this is:

$$\Delta w_{i,j} = \eta g(x_i(t), t_i(t)) h(x_j(t), w_{ij}) \tag{1.13}$$

Here, g is now a composite function - corresponding to the prior/pre-synaptic neuron (i)- and h is its a composite function corresponding to the post-synaptic neuron (j). This equation will be re-iterated in the Section 1.3.5 shortly.

However, these formations are not ideally adapted if we take into account the precise, rather than general, timings of spikes. The Spike-Timing Dependent Plasticity, or STDP model, then refines the Hebbian model on a more precise time scale; biophysically, the relative timing of the pre and post-synaptic spikes relates to changes upon voltage or conductance of the membrane of the focus neuron [110]. STDP is far richer than Hebbian learning because it provides a local, unsupervised engine for detecting temporal correlations, or anti-correlations, which

may provide a window into the casual origin of learning or inference. Since the discovery of STDP, the model has been discovered to perform unsupervised clustering of sequences [104] and visual features [111], as well as to animate competitive learning between neuro-circuits [112]. Interestingly, STDP may also approximate Bayesian computation [113, 114].

1.3.4 Perceptron: The Classic 'Neurocomputer'

In 1958, Frank Rosenblatt proposed the first network-based hypothesis for neuronal learning (classification), the 'perceptron'. This scheme is often referred to as the fundamental linear classifier of machine learning [115]. Rosenblatt abstracted from basic Hebbian principles of association by weight, corresponding alteration in synaptic strength Δw proportional to the firing of these two neurons together, and proposed a mechanistic model for behavior in which n inputs from pre-synaptic neurons $x_1, x_2, \cdots, x_n$ are integrated into a single output signal, y given as follows:

$$y = \varphi(T) \tag{1.14}$$

$$T = w_0 + w_1 x_1 + \cdots + w_n x_n \tag{1.15}$$

where T denotes the weighted sum of input signals that has been offset by a correction factor w_0. The φ function works as follows: it returns 1 if $T > 0$, else, returns 0 (this is often referred to as the Heaviside function). Note that, while the thresholding operation and binary output resembles neuron behavior in a conceptual sense, the perceptron model is considered a *non-spiking* algorithm since neuron outputs are timeless.

The learning operation involves an iterative procedure where a vector of weights $\boldsymbol{w}$ is updated following the output φ given input vector $\boldsymbol{x}$ for each example. By construction $\boldsymbol{x}$ takes values from $\mathbb{R}^n$. This space is divided into two subspaces $\mathbb{R}_1$ and $\mathbb{R}_2$ ($\mathbb{R}^n = \mathbb{R}_1 \cup \mathbb{R}_2$ and $\mathbb{R}_1 \cap \mathbb{R}_2 = 0$) so that the desired output (training task) $\hat{y}$ behaves as $\hat{y} = 1(0)$ for $x \in \mathbb{R}_1$ ($x \in \mathbb{R}_2$). The update to w is done as follows:

$$\Delta \boldsymbol{w} = \begin{cases} \eta \boldsymbol{x} & \boldsymbol{x} \in \mathbb{R}_1, T < 0 \\ -\eta \boldsymbol{x} & \boldsymbol{x} \in \mathbb{R}_2, T > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.16}$$

Here η represents a suitably chosen analog damping factor (often called "learning rate"). Updates are performed until a convergence is reached or the program halts. Convergence is guaranteed if and only if the set of inputs (the training set) is linearly separable; that is, if the positive and negative examples (recognized and non-recognized) can be separated into two different groups by a plane (or, more accurately, an n-dim hyperplane) [116]. Famously, the perceptron model cannot learn the XOR logic gate [117]. However, even if the system is not guaranteed to converge, it will still do its best to approximate a (linear) solution to a non-linear problem.

Derivation of LMS Algorithm

Let $y^p = \sum_{i=1}^{n} w_i x_i + b$ be the actual output of a given neuron in response to $\boldsymbol{x}$, where b is an offset or bias. Then, for a single example p, the error is:

$$E_p = \frac{1}{2}(d^p - y^p)^2$$

given the mean-squared error formulation, assuming d^p is the teacher signal or expected value. In addition, E_{tot} can be derived for all presented samples $1 \dots P$:

$$E_{\text{tot}} = \sum_{p}^{P} E_p = \sum_{p}^{P} \frac{1}{2}(d^p - y^p)^2$$

The overall problem set-up is to minimize error $min(E_{\text{tot}})$ by transforming $\boldsymbol{w}$ into $\boldsymbol{w_\theta}$, if $\frac{\partial E_p}{\partial w_\theta} = 0$ Classically, we achieve this optimization problem by adjusting each weight after every new sample has been presented as follows:

$$\Delta w_j = -\gamma \frac{\partial E_p}{\partial w_j}$$

Where γ is a constant, often called the learning rate. However, we still need to obtain the derivative of the samples' error and every weight's w_j contribution to it ($\frac{\partial E_p}{\partial w_j}$). To do this we can separate it out into the contributions related to the output and input:

$$\Delta w_j = -\gamma \frac{\partial E_p}{\partial y_p} \frac{\partial y_p}{\partial w_j}$$

Since the derivative of the cost-function with respect to the output can be easily obtained,

$$\frac{\partial E_p}{\partial y_p} = \delta^p = -(d^p - y^p)$$

and $\frac{\partial y_p}{\partial w_j} = x_j$, we finally obtain:

$$\Delta w_j = \gamma \delta^p x_j$$

1.3.5 Adaline: A Generalized Perceptron

While the perceptron makes discrete computational adjustments, the Adaline is a more general and powerful extension introduced by Bernard Widrow [118] which considers the difference between the output (prediction) and the expectation (teacher) in the execution of the learning step. Explicitly, given the network's prediction (output) $\hat{y} = \boldsymbol{wx}$, where $\boldsymbol{w}$ are the present weights and $\boldsymbol{x}$ is the presented example, a loss of $(\boldsymbol{y} - \hat{y})^2$ is incurred, where $\boldsymbol{y}$ is the set of

true/actual values (eg labels). Then,

$$\Delta \boldsymbol{w} = -\eta(\boldsymbol{y} - \hat{y})\boldsymbol{x} \tag{1.17}$$

In the literature, this is variously referred to as the 'delta rule' or the least mean squares (LMS) algorithm [119]. Note that the adaline weight update rule is the same as that used for larger multi-layer networks. The derivation for this is shown in 'Derivation of LMS Algorithm.'

One interesting about the delta rule is that it could also be directly derived from the Hebbian with teaching formulation (Eqn. 1.13) when h is simplified to x_j, and g is simplified to $\eta(t_i(t) - a_i(t)$, as above. In this sense, this simple rule is considered not only a first-order approximation of gradient descent, but also a first-order approximation of simple reinforcement learning [120].

The schematic of a simple Adaline is shown in Fig. 1.2 (a), where the input $\boldsymbol{x}$ is fed through a set of weights $\boldsymbol{w}$ and taught by the simple teacher signal t, which would produce one of two binary outputs that is compared to the sign of the output y. In Fig. 1.2 (b) , the multiple or parallel adaline construction is introduced. In this scenario, several adalines learn in parallel; they collectively produce an output $\boldsymbol{y}$, and each component learns relative to its component of the global teacher signal $\boldsymbol{t}$. In the context of a multiple-classification problem where one vote is counted, when a final inference is made, the 'winning' output or neuron y_* is typically computed as

$$y_* = argmax(\boldsymbol{y}) \tag{1.18}$$

Like perceptrons, adaline systems at best construct linear classifier boundaries.

1.3.6 Multilayer Perceptrons (MLP)

This problem of linear separability was the motivation for the introduction of a multi-layer perceptron model to deal with arbitrarily complicated input values. The essential elements of such a network are an input layer, a hidden layer of N internal nodes or 'neurons', and an output (or read-out layer). Hidden neurons solve the linear separability problem by changing dimensionality. In particular, given the N nodes a mapping to a representation in $\mathbb{R}^N$ space is made. Thus, with the introduction of hidden nodes, a multi-layer-perceptron (MLP) is now capable of mapping any continuous given input function to any degree accuracy- obtaining Turing universality. However, this requires an arbitrarily large hidden layer, which is often not practically realizable [121].

The network is feed-forward, *i.e.*, values propagate from input to output through the network. However, since the error function is only given to the ultimate layer, earlier layers are adjusted 'backward' in order to accurately approximate the target function. In this sense, the back-propagation approach is a spatial (layer-by-layer) extension of the general stochastic gradient rule (delta rule) already demonstrated.

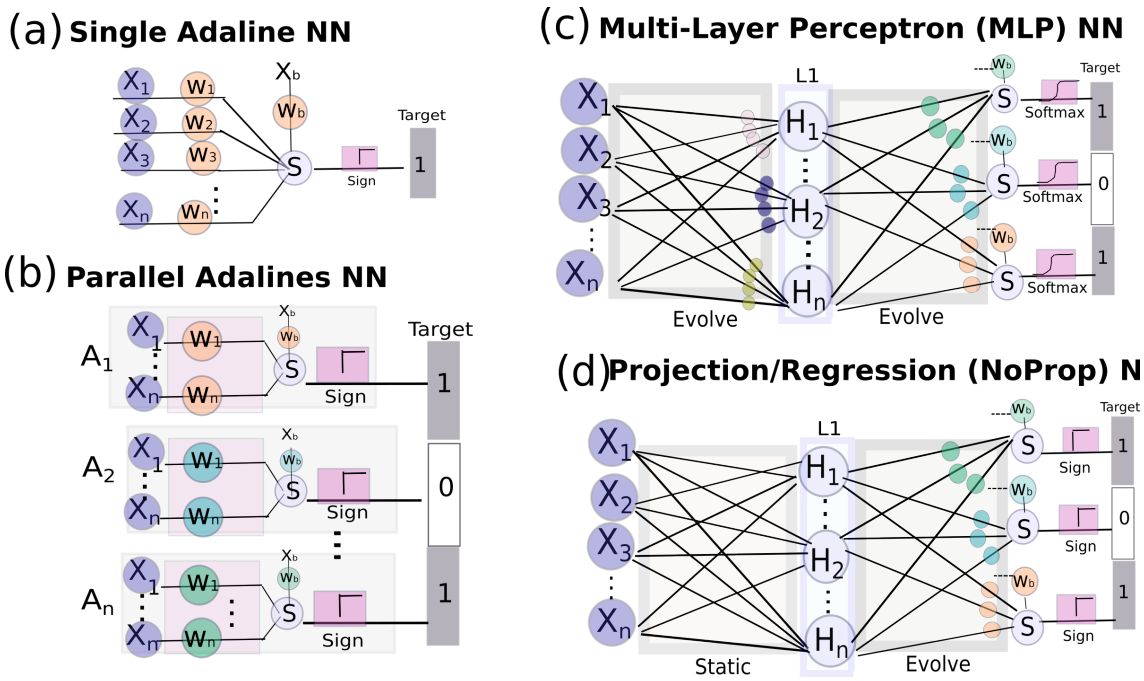


Figure 1.2: Conceptual depictions of the structure of supervised learning ANN models: Adaline, parallel adalines, MLP, and ELM/NoProp. In every model, some set of inputs $\boldsymbol{x}$ are mapped into a vector $\boldsymbol{w}$ (a) or a matrix of weights W (b-d), and in every cases a target function $\boldsymbol{t}$ directs the network's adaptation. Learning is performed in each case according to the methods described in more detail in the text.

1.3.6.1 Derivation of MLP with Standard Loss

Given a set of D training examples, the objective of such a network is to minimize the aggregate prediction error over the entire set squared error over the entire set. Prediction error for an individual is again difference between expected t_d and actual output o_d. If the loss or cost function is least-squares, this is

$$E[\boldsymbol{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \tag{1.19}$$

This global error function may be parameterized with respect to the vector $\boldsymbol{w}$ of all synaptic weights, of length n, as a collection of partial derivatives:

$$\nabla E[\boldsymbol{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right] \tag{1.20}$$

All weights in the first matrix/layer (input index i, hidden index j) update as:

$$w_{i,j} \leftarrow w_{i,j} + \Delta w_{i,j} \tag{1.21}$$

where

$$\Delta w_{i,j} = \eta \delta_j x_{i,j}. \tag{1.22}$$

As given before in Eqn. 1.17, η is a learning rate and $x_{i,j}$ is the pre-synaptic input for that particular synapse. While we immediately have $x_{i,j}$ (the network's input), *we cannot immediately obtain* δ_j, or the error derivative on the hidden layer. To obtain this, error must be back-propagated from the ultimate layer of K neurons (index k) to the first. Again following the chain rule, derivative of error is:

$$\delta_k = (t_k - o_k) f'(o_k), \tag{1.23}$$

where $f'(o_k)$ is the derivative of the activation function of the final layer. Once obtained for all output neurons, the ultimate error is propagated to preceding layers; for hidden layer neuron j:

$$\delta_j = f'(h_j) \sum_N \delta_k w_{j,k}, \tag{1.24}$$

here h_j is the activation of that middle layer neuron, and $f'(h_j)$ its derivative. This sum or rank-2 update is used to propagate error backward. Then, every synapse within this layer adapts following an edited version of Eqn. 1.21,

$$w_{j,k} \leftarrow w_{j,k} + \Delta w_{j,k} \tag{1.25}$$

where

$$\Delta w_{j,k} = -\eta(\delta_k x_{j,k}). \tag{1.26}$$

Finally, with the hidden layer gradients obtained by Eqn. 1.24, weights can be updated in the first layer as well. Thus, layer-by-layer learning is an algorithmic constraint in multi-layer systems using back-propagation. A visual depiction of an MLP is shown in Fig. 1.2 (c).

1.3.6.2 Improving MLPs with Info-Theoretic Cost-Functions

The process of backward-propagation can introduce problems that may not have been a factor with simple one layer/adaline learning systems. The system needs to traverse an increasingly complex gradient landscape, increasing the odds that a local rather than a true global minima may be selected; in addition, due to the large number of free parameters, the system may struggle to converge (find this optimum). Avoiding local/false minima (incomplete solutions) and speeding convergence is an active field of research in machine learning, and many viable partial solutions have been proposed. One particular topic will be discussed here since it is relevant to one of the later systems introduced: an improvement in cost-function chosen in the ultimate layer. The costs function is a critical constraint, since it determines which errors are being back-propagated to all previous layers

When squared error/MSE, also often-called L2 loss, is combined with a linear output function ($f'(O_k) = 1$), $\delta_k = (t_k - o_k)$ results, which is the delta rule. Although simple, the fragility of this approach has been demonstrated in the context of multi-layer software neural nets [122].

In particular, performance is overly dependent on initial conditions, and especially in multi-class classification tasks, backpropagated error often vanishes over several epochs; in both cases, slow or no convergence results. In contrast, the cross-entropy (log-loss) cost function is not as susceptible to these effects, as it always scales adjusted error according to the logarithm of difference between the expected and actual probability distribution [123]. The cross-entropy error function in a multi-neuron context, is defined for a given neuron k as:

$$L_{\text{CE}} = -\sum_{k} t_k log(o_k) \tag{1.27}$$

Where t_k is the target distribution (label value for that neuron) and o_k is the actual distribution (output). In batch form, or accumulating cross-entropy loss over S total samples (index $i = s$) at neuron k (total K) is:

$$E = -\sum_{i=s}^{S}\sum_{k}^{K} y_{ik} \cdot log(o_{ik}), \tag{1.28}$$

Now, let us assume that the immediate output o_k is transformed into the softmax output s_k as:

$$s_k = \frac{\exp^{o_k}}{\sum \exp^{o_i N}}, \tag{1.29}$$

Softmax has some intriguing properties, foremost, that it outputs a probability distribution of the outputs relative to each other which always falls in the range of reals ([0,1]); this prevents vanishing or exploding output issues over multi-epoch training periods. Indeed, the literature suggests that a softmax transformation of original network outputs can bestow some immunity to convergence failure [124]. Typically, the derivative of softmax is rather complex:

$$\frac{\partial t_k}{\partial s_k} = \begin{cases} s_k(1-s_i) & if\, i = k \\ -s_i.s_k & if\, i \neq k \end{cases} \tag{1.30}$$

However, the combination of softmax and cross-entropy functions produces a simplified form that substantially reduces this complexity. The derivative of the cross-entropy loss function with regards to softmax output s_k is given as:

$$\frac{\partial L_{\text{CE}}}{\partial s_k} = -\sum t_k \frac{1}{s_k}\frac{\partial t_k}{\partial s_k} \tag{1.31}$$

Given the cases earlier defined for $\frac{\partial t_k}{\partial s_k}$ in Eqn. 1.30, and assuming that the vector of labels $\boldsymbol{t}$ is using one-hot encoding (expected = '1', *e.g.* $k = i$, and unexpected = '0', *e.g.* $k \neq i$), Eqn. 1.31 substantially simplifies:

$$\frac{\partial L_{\text{CE}}}{\partial s_k} = s_k - t_k \tag{1.32}$$

This is effectively as simple as the L2 case (delta rule), while gaining a far richer (probabilistic) cost function. This can help immensely in a multi-layer gradient learning environment.

1.3.7 Random Projection (ELM/No-Prop) Learning Systems

Another approach to learning in multi-layer systems that also allows for universal solution to non-linearly separable problems is to treat a certain set of synaptic weights as un-changeable, often those in earlier layers, and to evolve (adapt) only the ones that come at later layers. This approach has been independently proposed by [125, 126] and is widely referred to as either 'NoProp' or the extreme learning machine (ELM) approach in the literature. Its successful implementation relies upon three pre-conditions:

- Earlier, static layers of the ANN have higher dimensionality (larger number of hidden neurons) than the relatively compressed read-out component.
- The large hidden layer is usefully transforming the input by performing some variety of non-linear projection (it could not be linear).
- These static layers must have random weights, which could for instance be chosen from a random Gaussian values, or could be made around another uniform distribution. Actually, the exact distribution does not matter, but what is critical is that these weights *do not change* during the training or testing phases. Indeed, the random field of synaptic weights/connections have to be treated as a constant by the learning system.

The state of the jth hidden neuron (of M total) is given by:

$$H_j = f(\sum_{i=1}^{N} W_{in,ij} X_i), \tag{1.33}$$

where f is a non-linear activation function, such as $tanh$, sigmoid, or some other choice. However, while f must be non-linear, it does not necessarily need to be differentiable, since the global error function does not need to pass through this layer. A toy version of a random projection network learning is demonstrated in Fig. 1.2 (d), where L1 performs the projection function above, the first layer of weights W_{in} is static, and the second layer of weights W_{out} is obtained either analytically or progressively according to an appropriate method for linear regression, e.g., a pseudo-inverse of the collected states from the hidden layer (if analytical is chosen).

In its original formulation, this ANN model does not possess much complexity at the hidden layer. However, the literature demonstrates recent improvements which may allow random projection learning machines to achieve competitive performance in some task varieties. Such 'enhanced-ELM" systems principally benefit from two additions: increasing complexity of system, and expanding the depth or breadth of layers.

1.3.7.1 Enhanced ELM/NoProp systems

Increasing Complexity

In [127], various improvements such as computed inputed weights (CIW), image pre-processing,

receptive fields, were discussed to collectively boost ELM performance to achieve only 1% loss on the MNIST test-set. All of these operations increased spatial intelligence on the task, but did not add temporal processing abilities. However, in [128] Tapson suggests adding various time-dependent synaptic kernels at the hidden layer (the kernels are particular mathematical functions). This gives the system intrinsic spatio-temporal processing ability; however, it is not recurrent. In parallel, Ortin et al. have suggested the closeness of the ELM approach and an approximation of true recurrent neural networks called Reservoir computers, when time-delays are added at hidden layer neurons [129]. More information on reservoir computers is provided in Section 1.3.9.

Increasing Depth

Another option is to increase system depth, defined as the number of hidden layers. While in large MLP systems backpropagation is done through every layer, in Deep ELM systems (D-ELM), hidden layers typically switch between random projection (static) and learned (evolved) in succession, such that each regression system's learning task is simpler. Two recent examples include the integration of hierarchy are shown in [130, 131]. Finally, many smaller projection modules may be connected together to access greater task performance. In [132], it was demonstrated that separate, small ELM auto-encoders collaborating on a learning task improved performance relative to a homogeneous single hidden layer of equivalent combined size.

1.3.8 Recurrent Neural Networks

So far, the ANN designs we have considered here have been either entirely feed-forward, or if they do use back-propagation, use this alternate movement only in the error-correcting step. However, in another set of ANN designs broadly known as 'Recurrent neural networks' (RNN), internally evolving state variables, *e.g.* those possessed by individual neurons/nodes in the network, are treated as a matter of intrinsic computation. In partcular, in an RNN, neurons send feedback to other neurons (or themsleves).

Thus, in an RNN scheme, communities of neurons form loops rather than just lines; complex feedbacks and circularities predominate over linearity. Moreover, activations of individual neurons (nodes) then also take on a degree of interdependence of memory in between themselves. RNNs includue a wide variety of computational models including, Hopfield models (in which all connections are symmetric) [133], simple recurrent neural network models such as the Elman or Jordan networks, and complex, bio-inspired competitive neural field models [134].

Recurrent systems possess a great deal of computational power including non-linear mapping (segregation) abilities, stored associative memory, and collective state behavior described by attractor dynamics. Moreover, since these systems are spatio-temporal in nature, they are

naturally suited to performance inference on complex, real-world tasks. For instance, a RNN was recently trained to not only memorize digits, but to transcribe (draw) them using a motor [135]. However, an online training scheme for such systems has remained elusive [136, 137].

A conceptual map of an RNN is illustrated in Fig. 1.3 (a), where the key distinctions between this system and the multi-layer perceptron, its closest cousin, are noted as the recurrent arrows mapping each hidden neuron's state H_j back into itself, along with the recurrence between the output nodes and the hidden layer conferred as the states evolve. Indeed, back-propagation must be applied not backward through layers but through the state of the network in time, a process often referred to as back-propagation through time (BPTT). BPTT is visualized in Fig. 1.3 (b), where the state of two hidden layer neurons are depicted over three discrete time steps. As visible in the small red arrows, standard RNNs learn by passing the gradient backwards over many time iterations, which both increases their computational power and complexity simultaneously. As for the second fact, this may lead to the 'vanishing gradient problem' [138]. One solution to this problem exists in the form of a new variety of memory cell called the Long-short term memory (LSTM) cell, which both allows for the network to evolve through time and preserves crucial information about the gradient [139]. While powerful, this design is famously complex. A related and slightly less complex is known as gated-feedback RNNs (referred to as the GRU cell) [140]. The two models are compared directly on several tasks in [141]. Instead of these models, we more closely study a related proposal which exploits the dynamics of recurrent neural networks without requiring backpropagation, as introduced in the following section.

1.3.9 Reservoir Computers

In 2002, Maass et al, described the information processing capability and a simple read-out (training) mechanism for recurrent neural circuits [142]. The readout point can receive input from hundreds or even thousands of different inputs and still effectively discriminate important information from this high dimensional transient set of states- a so-called Liquid State Machine (LSM). Independently, Jaeger et al derived a similar formulation (Echo State Network or ESN), which concerns ANNs that conduct more processing within their dynamical reservoirs [143]. Together, we refer to these equivalent models as reservoir computing (RC) systems, which use the combination of random weights and recurrent projection spaces- often called the liquid state- to generate rich outputs that can be used in spatio-temporal classification or processing tasks. Quite similarly to ELM/NoProp networks, these systems typically accumulate a large set of activations from the un-trained part of the system and then use ridge regression to analytically obtain output weights mapping these complex outputs to the desired target functions. Subsequently, the system can be used for classification or clustering tasks. Methods for training reservoirs online also exist, including of particular note the FORCE algorithm [144] which nudges reservoirs towards edge-of-chaos stable states.

In an LSM, given $\boldsymbol{u}$ as the input set, and an operator known as the liquid filter L^M (the

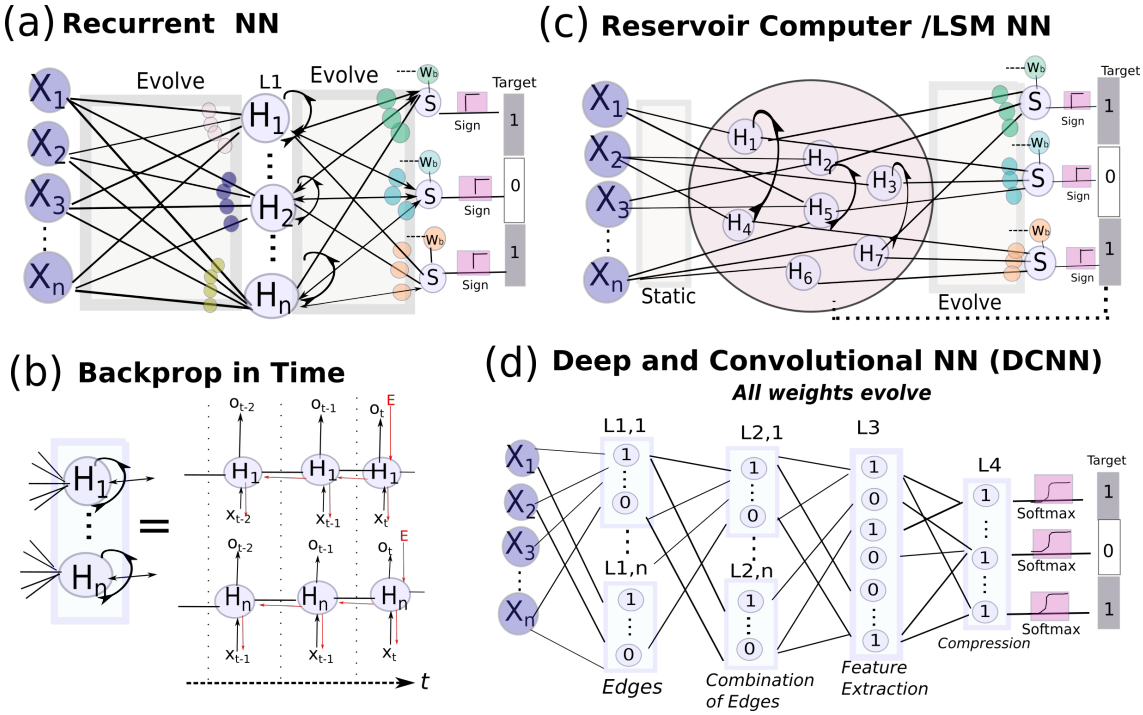


Figure 1.3: Conceptual depictions of RNN, Reservoir Computer, and DCNN computation models. In every model, some set of inputs $\boldsymbol{x}$ are mapped into a set of matrices of weights $W_1 \cdots W_n$, and a simple target function $\boldsymbol{t}$ is used to direct the network's adaptation. Panel (c) highlights the temporal evolution of the RNN's hidden layer.

function performed by the dynamics of the reservoir on the input), a transient or liquid state $x^M(t)$ is obtained at some time t. This liquid state is visualized as the central circle shape with many internal recurrent nodes in Fig. 1.3 (c). Finally, f^M represents a memory-less readout that transposes this current state/value, into the output. While the liquid filter is 'set', it is the choice of the readout function or map that directs the processing or computation conducted within such a recurrent network. Just as the MLP, the LSM model is computationally universal (can compute anything a Turing machine could).

A LSM derives this power from two necessary and sufficient properties: the separation property, and the approximation property. The first describes formally the separation between the trajectories of internal states in response to an external stimulation, as ripples would propagate on a pond; the latter property describes the ability of the read-out mechanism to accurately distinguish these internal states into the target output(s). In addition, the computational power of this network is strongly affected by its fading memory, which is the ability to deduce from the value at $x(t)$ information about the inputs at some previous times (from $u(t-1)...u(t-T)$), but not all previous ones- hence fading. In this way, the reservoir is both entrained to the input and also flexible enough to continue adapting through time. This property, also typically referred to as the Echo State Property (ESP), was demonstrated in [145] to depend not only on the state and design of the reservoir, but also the format of the input. The ESP is

somewhat the inverse of another key metric of a RC: its memory function. This function, $m(t)$, distinguishes how many steps backward in time the present state $x(t)$ could be reconstructed from; its integral is the system's memory capacity.

The two formal properties described earlier (separation and approximation) can be observed in physical systems that perform effective spatiotemporal projection/pre-processing. In one case, a literal 'liquid machine' was built [146]. In this pedagogical experiment, a bucket of water implements perfect learning of the XOR function and noisy but respectable performance on digit recognition when these inputs are projected into the water as ripples and fed to a classifier (performance). Additionally, recent work in experimental neuroscience suggests that the LSM may be an appropriate model for at least one sub-system of the brain- the cerebellum [147]. Given all these properties, the RC/LSM model proves to be a relevant schema for investigating the brain itself. A recent study indeed used RC computational methods to better understand the dynamics of the fronto-striatal system [148].

1.3.10 Modern Deep and Convolutional Approaches

In the 1980s, Fukushima proposed the integration of small receptive field computing kernels into a larger, and later hierarchical architecture, integrating each of their individual encodings [149, 150]. Subsequently, LeCun demonstrated the power of these deep, convolutional image processing in image processing tasks using his LeNet architectures [151]. While these powerful architectures were conceptually ready, they lacked both the hardware to be properly executed, as well as the large, well-labeled datasets necessary to efficiently generalize on difficult tasks. Between 2005-2012, the triple combination of successful 'tweaks' to increase learning efficiency, the adaptation of existing hardware (graphical processing units [GPUs]) to massively increase training speed, and the development of well-designed datasets opened the doorway to a renaissance in artificial intelligence. Demonstrating the last point, in 2012 a deep neural network called AlexNet was able to perform multi-class image recognition on a difficult image dataset called ImageNet at remarkable levels of performance [152]. Since then, improvements to these sorts of deep networks have resulted in super-human level accuracy [153]. We call these modern, complex networks deep and convolutional neural networks (DCNNs). Beyond image recognition tasks, DCNNs have been adapted to achieve remarkable performance in audio (speech) recognition, drug-discovery, material research discovery, and a variety of other tasks [154]. Although the field still lacks a principled theoretical understanding as to why these powerful networks are generalizing so well, what is well-understood is that the powerful computational abilities of these networks largely derive from the ability of successive non-linear encodings (layers) to capture structures (correlations) in extremely high-dimensional data. Exact solutions to similarly deep linear neural networks have shown that, while they can be exactly compressed (combined) into an equivalent shallow network, no such decomposition can be made for deep, non-linear neural networks [155]. A toy illustration of a DCNN is shown in Fig. 1.3 (d), in which two convolutional layers early in the network detect and combine edges, and

later layers combine and compress key features. The last weight field (between L3,L4) is here fully-connected and performing the final classification prediction is typically obtained using the softmax function (Eqn. 1.29).

Deep neural networks, while powerful, have some critical constraints. Even with the advent of new GPU hardware and accelerated training, for two fundamental reasons these networks are very slow to learn and extremely power hungry:

- In order to correlate extremely high-dimensional features, DCNNs need dozens of layers. As a result, some large DCNNs contain dozens of layers, millions of neurons, and hundreds of millions of individual synapses.
- The backpropagation algorithm must proceed layer-wise, rather than simultaneously training all layers, as some un-supervised or energy-based models might.

As a result, these models can take days or weeks to compute in server clusters, making them mostly inappropriate for local, off-grid learning systems. Some effort has been made to reduce the complexity of the elements of these ANNS; they are often built with the simplest possible differentiable neuronal units such as rectifiers [153], and reduced precision weights and activation functions are also being considered to accelerate their convergence (training) [156, 157]. In addition, a deeper conceptual understanding needs to be unlocked in order to prevent DCNNs from being mis-used in industrial or safety-oriented contexts due to the well-known susceptibility of these types of networks to subtle perturbation during either inference (testing) or training stages (e.g., adversarial examples meant to break the network). As noted in [158], DCNNs learn to recognize classes in an inherently discontinuous way, which may make them intrinsically susceptible to these sort of attacks.

1.3.11 Evaluation: implementing a vision task on ANNs

In this section, we demonstrate the power of these type of networks as well as briefly compare their performance on an equivalent task. The chosen vision classification task, which will be referred to and used several times in the upcoming works as a general test of the ability of the neurmorphic ANN designs we propose, is the M-NIST task of handwritten digits. The M-NIST task was reconstituted from the original, far larger NIST databases of handwritten digits obtained from approximately 250 writers; in its modern form, this database consists of 60,000 training examples and 10,000 testing examples of the ten digits, each example being a grey-scale image of 784 pixels [159]. Individual examples of each of the digit classes in the database are visible in the left part of Fig. 1.4; as pictured, these images are typically 'unwrapped' and presented to the input layer of the given ANN system at once.

Next, we briefly demonstrate the absolute as well as comparative power of the ANN structures introduced in the previous sections on this task. The following benchmark results were all obtained from the benchmarks listed at [1] with the exception of the NoProp and Reser-

[1] http://yann.lecun.com/exdb/mnist/

ANN type	Performance on Test Set	Parameter Notes
One-layer adaline	88%	MSE
Multi-layer perceptron	95.3%	$M = 300$, MSE
Multi-layer perceptron	98.5%	$M = 300$, softmax + cross-entropy
NoProp	96.7%	$M = 3500$
RNN (LSTM)	99.05%	$M = 128$ units
Reservoir	96.3%	$M = 500$ nodes in liquid
DCNN	99.25%	Many hidden layers (Le-Net 5)

Table 1.2: Test results obtained on the MNIST Test set for all the considered ANN structures. M is the number of hidden layer neural or memory cell components.

voir system results, which were obtained using extensions of the TensorFlow [160] and Oger [161] Python packages, respectively. As visible in Table 1.2, the one-layer system is a notable poor performer, as it can only construct linear mappings on the task. Among the complex multi-layer systems, the convolutional neural network demonstrated in [151] is most performant, while the multi-layer perceptron (MLP) systems with complex cost function and the RNN system (using LSTM memory cells) obtain third and second place, respectively. However, as demonstrated in the major difference between the two equivalent MLP architectures, not only the architecture but the choice of the cost function can have a major effect on the network's ability to generalize.

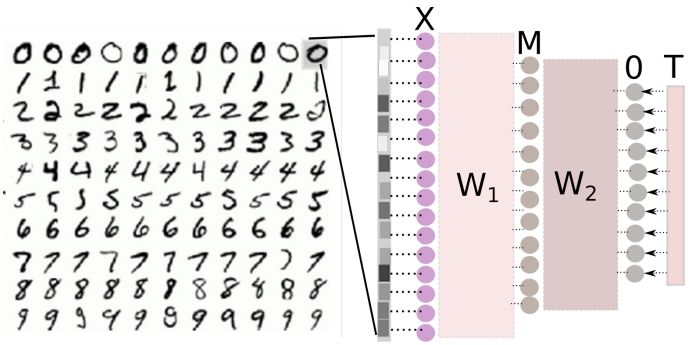


Figure 1.4: In the left, many training examples of handwritten digits are illustrated. In the given moment, one of them belonging to the class '0' has been selected, unwrapped and presented to a set of input neurons $\boldsymbol{X}$. After presentation, outputs $\mathbf{0}$ are compared to the target values or label, T. The generic ANN pictured is a multi-layer perceptron.

1.4 Learning with memristive nanodevices

1.4.1 Uniting the Strands: computation meets nanodevice

This penultimate section stands as bridge between the two previous sections, namely, the device perspective and the algorithmic perspective of neuro-inspired computing. As noted in

[27], memristive devices are an ideal building block for future neuro-synaptic hardware learning systems, due to their simple (non-volatile weight storage) as well as complex (ability to emulate spike timing effects) physical behaviors. In addition to neuro-inspired synaptic functions, memristive devices are chemical computing components that may be combined in more complex ways to realize molecular turing machine or logic gate architectures [162].

In the following sections, we discuss a variety of elementary learning or adaptation strategies so far considered with emerging memristive devices. Emerging memristive devices playing the role of nanosynapses may be used for systems trained "off-line" (or *ex-situ*), used only for inference. However, their most attractive use would be in learning capable systems, which can learn data "on-line" (or *in-situ*) without reliance to the cloud or external computers [163]. Due to this work's emphasis on online or local learning, off-chip examples are very briefly introduced, while a great deal of emphasis is placed on various online computing strategies and algorithms.

1.4.2 Off-chip learning

In off-chip learning, weights are obtained using another generative or training model, and then imported directly to the crossbar system for the inference operations. This approach is further compared and contrasted with on-chip learning in [163]. This sort of approach can make sense when the weights correspond to a very complex and difficult to train model, e.g. a large convolutional neural network. An example of an HfO_2 learning network that primarily uses imported weights is given in [166].

1.4.3 On-chip learning

In the online learning approach, neuromorphic architectures employ the intrinsically adaptive behavivor of devices to realize learning [164]. In general these architectures have a surprising degree of immunity to imperfect devices [165], although these tolerances may depended greatly on the specific memristive device model (or broad 'family' of device).

There are two varieties of on-chip learning: supervised and un-supervised. In on-chip supervised learning, the system has access to a teacher signal and/or cost-function; in the context of a classic data-science task such as classification, these would be labels for classes. Such a teacher/loss function is critical to the performance of such systems. In contrast, semi-supervised or un-supervised systems learn without direct knowledge of what is 'right' and 'wrong'; in its place a variety of local dynamics and plasticity effects nudge the network towards a native understanding of the data being fed into it.

1.4.3.1 Plasticity-based learning rules

Standard STDP modulates weights according to $\Delta t = T_{pre} - T_{post}$, as in Fig. 1.5 (a). A binarized version of this learning rule, as depicted in Fig. 1.5 (b), does not analogically change weights,

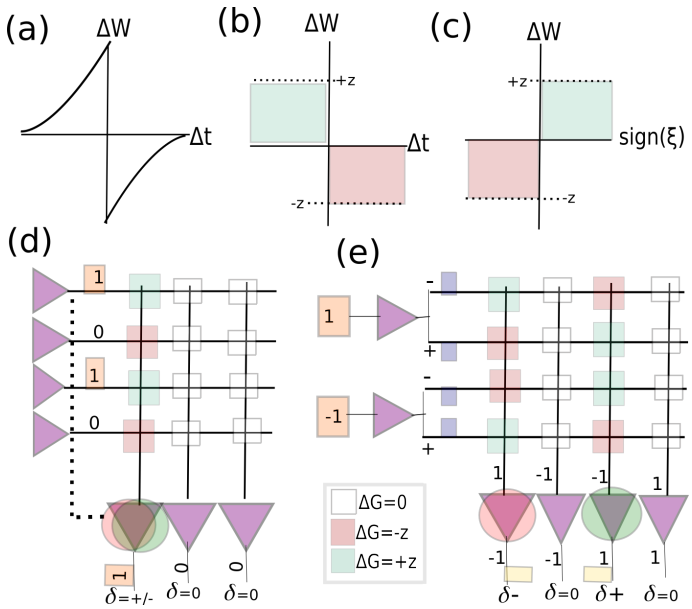


Figure 1.5: (a) Demonstrates the typical analog STDP curve; (b) shows a binary approximation to it; (c) shows the equivalent binary representation when timing is encoded by the expectation signal, δ. (d) Demonstrates how an un-supervised crossbar learning system adapts its weights according to (b) ; (e) Demonstrates how a supervised crossbar learning system, now with double the devices to encode negative weights, adapts its weights according to (c).

but would merely update a fixed amount according to $sign(\Delta t)$. This system can be physically implemented using a crossbar of nanodevices in which a post-synaptic firing using a winner-take-all scheme (the firing or 'winning'). As illustrated in Figure 1.5 (d), this scheme punishes the in-active (anti-correlated) synapses on that active output line, while rewarding active (correlated) ones; all synapses connected to the inhibited neurons are un-affected. Note that, to physically be implemented in a crossbar, this scheme requires a 'back-propagated' spike along the pre-synaptic lines in order to program the devices successfully (dotted line). It also requires careful tuning of the output neuron's parameters (for instance, to avoid one output neuron always firing). Further implementation details of this particular scheme are shown in [73].

In 2010, Pershin and DiVentra built a memristor emulator using standard electronic components, and used it to build a circuit which demonstrated a simple correlation or conditioning problem often known as 'Pavlov's dog' (correlating the 'sound' or 'sight' of food to salivation) [167]. A more complex scheme was demonstrated with actual filamentary resistive RAM [27], in which a crossbar of silicon nanowires with amorphous Ag filaments exposed to various Pre and Post synaptic signals were shown to more explicitly match the canonical experiments [110]. Since these canonical demonstrations, a variety of other demonstrations of STPD using memristive devices have been attempted or claimed; most use special voltage wave-forms or companion circuits to obtain the results [168]. In analog mode, phase-change memory devices have realized Hebbian learning [169], while in binary mode, PCM devices have been used to

realize a low-power neuromorphic engine for car-counting on a highway [170]. Another binary neuromorphic circuit design using these devices was given by [171], who notably managed to use the stochastic aspects of this device as a feature rather than a bug for learning.

Application to semi-supervised or unsupervised learning

Bio-inspired approaches using spike-timing dependent plasticity (STDP) aim to intrinsically identify correlations in input data, an operation which may pave the way towards a generative system that learns without (un-supervised) or with just a few (semi-supervised) training labels. Notably, in [172], a single spiking layer with 300 output neurons achieved 93.5% classification on the M-NIST task without training labels. While such a scheme relies heavily upon complex behavior in output neurons, this demonstrates the ability of nanodevices to perform unsupervised clustering. An unsupervised system reaches 95% test-set classification, also on M-NIST, in [173]. However, reaching this performance requires over 6000 inhibitory/excitatory spiking neurons and fine tuning of many parameters. Hardware spiking implementations using memristive devices as nanosynapses have been realized, identifying simple images [174]. In [175], a 1T1R CMOS neuron/PCM synapse circuit realizes an artificial STDP dynamic; individual classes from the MNIST task are memorized with this dynamic. Recently, a fully spiking network using STDP was implemented in a crossbar of memristive devices in [176].

Physical STDP-like behavior

In most of the above cases, the STDP or STDP-like phenomenon did not derive from the physics of the nanodevices implementing it, but instead arose more as a systems-level effect. However, other examples have demonstrated STDP effects arising from material physics. For instance, the nanoparticle computing device earlier mentioned intrinsically demonstrated spike-timing dependent plasticity (STDP) effect [177]. A recent, truly intrinsic use of STDP in more scalable ECM ionic silver filamentary devices has been explored in [178]. The complex plasticity effects of this device are also explored in more detail in this thesis in Section 4.2. These developments are exciting, since intrinsic plasticity mechanisms with nanodevices may pave the way to new types of sparse or spiking neural networks with minimal energy draw [164].

1.4.3.2 On-Chip Supervised Learning

Mapping delta rule to a crossbar

In the delta rule, sign-symmetric gradient learning is employed to optimize the network such that it performs a certain task or mapping function (e.g., classification on pixels). One way of understanding the dynamic implementation of this algorithm in a crossbar environment is to track a local error parameter, $\delta = E_{\text{val}} - O_{\text{val}}$, instead of the global error function. In words, the difference between the expected or teacher signal and post-synaptic (actual) output determines whether the neuron should adapt or not. When a binarized version is implemented, two archetypal adaption cases are possible; if $sign(\delta)$ is positive, the neuron is not as active (posi-

tive) as it should be, and so a reward signal to increase its output is backpropagated; conversely, if $sign(\delta)$ is negative, the neuron is not as quiet (negative) as it should be, and so it is punished (weight decreased). However, for each output neuron there is also a dependence on the pre-synaptic values X_j that led to that case. If we define $\xi = sign(X_j)sign(\delta)$, we obtain a simple binarized 'STDP-like' rule (Fig. 1.5 (c)). Inspecting the schematic of crossbar implementation of this simple rule (Fig. 1.5 (e)) the only additional complication is that pairs of devices are differentially programmed in this approach since we need two device to encode a full range of weight $G_{\text{syn}} = (G_{\text{pos}} - G_{\text{neg}})$. In other words, two nanodevices are sufficient to encode one nanosynapse with a full range of weight/response. Thus, inspecting the crossbar again one can see that the $sign(\xi)$ rule is also valid for all devices on the positive lines, and sign-inverted on the negative lines. Furthmore, the comparison of the delta rule and STDP are evident when considering this scheme. In particular the synapses connected to δ_+ case neurons fulfill Hebbian learning (connected/active rewarded and inactive punished); the δ_- connected synapses undergo the opposite, or 'anti-Hebbian' learning. This formulation yields an elementary relationship to models of dendritic learning; in [179], dendritic learning is modeled as a system which adapts and is punished or rewarded corresponding to the difference between its expected and global contribution to the global error. According to this interpretation, single neurons employing multiply adaptive dendrites aim to minimize global surprise.

Experimental implementations with nanodevices

Due to its simplicity and universality, this rule or slight variants of it, has been the staple of many simulated and experimental works with memristive devices [81, 180–183]. For instance, single layer experimental demonstrations of this principle have been made with memristive oxide devices [163, 181]; recently, a multi-layer prototype using this learning strategy was also realized [184, 185]. This work used PCM devices in the analog mode to realize more elaborate multi-layer systems performing the M-NIST machine learning task already described in Sec. 1.3.11.

1.4.3.3 PCA or sparse coding with memristive devices

Oja introduced a theory which suggests that a neuron can behave as a principle component analyzer (PCA) [186]. Subsequently, Sanger expanded this result into a general algorithm called Generalized Hebbian learning algorithm, often also referred to eponymously as Sanger's Algorithm [187]. Recently, these principles were used to experimentally perform un-supervised clustering with a crossbar of memristive devices [188].

A more complex, full-featured bio-inspired unsupervised algorithm is the sparse coding method described by Rozell [189]. In effect, this algorithm constructs an over-complete set of basis vectors to represent input vectors (this is in contrast to the PCA approach, which only creates a complete set). A variety of works have aimed to the realization of this concept with nanosynapses, the most recent and notable being the full implementation of sparse-coding in

a crossbar environment where it constructed dictionaries to reconstitute small images [190].

1.4.3.4 Recurrent or reservoir learning systems with memristive devices

Limited recurrent networks in the classic Hopfield configuration can realize classic associative memory abilities. Recently, experimental works have demonstrated associative memory built with PCM nanosynapses [169]. In [191] the authors construct a more elaborate reconfigurable associative memory architecture. A more classic RNN-inspired design using memristive devices in crossbars was proposed and simulated in [192], but to our knowledge full RNNs have not yet been built with memristive devices.

Reservoir inspired schemes built with memristive devices were first mentioned in [193] and simple classification task were attempted in this context. Subsequently, more complete designs have been proposed but they used memristive devices only in the read-out layer [194]. In [195], a hierarchical approach exploiting the dynamics of smaller memristive reservoirs themselves was proposed, but the optimal read-out and training scheme for such a complex system remains an open question. As it is, all schemes using memristive devices for online learning can exploit a favorable fact about reservoirs, which is that they are intrinsically good at processing time-series data. While all reservoir computers are dynamical systems that can perform intelligent mappings on inputs (act as a filter), not all dynamical systems are necessarily reservoir computers [196]. In [197, 198], attempts were made to understand how basic memristive systems can act as intrinsic online computing filters, and upon what set of input signals they may be effective in performing this operation.

1.4.4 Physical implementation of synaptic arrays

1.4.4.1 Grid (Crossbar) Topology

Crossbar topologies have been successfully implemented since 2003 [44] due to the ease of fabrication using e-beam lithography and photo-lithography processes, the potential for ultra-dense integration, and the ability to easily co-integrate such a structure with other CMOS layers [45]. In such a configuration, a large grid of top-electrodes contacting a bottom grid of electrodes (typically both nanowires), giving rise to a massive number of electrical switches. Specifically, given m top wires/electrodes, and n bottom, there would be nm total. The two most popular variants on this fabric relate to the way that individual device, or groups of devices, are accessed for reading and programming within this synaptic matrix.

The first, known as the passive (1R) architecture places no additional device at every cross-point, but still employs control circuitry at one or both periphery of the fabric. Early experimental results demonstrated the possibility of a deleterious sneak-path effect, in which current flowing through devices (notably in the ON or low resistance state) disturbs the states of surrounding devices [199]. As a result, an alternative crossbar design known as one-transistor-one-resistor (1T1R) was designed; it places an additional transistor CMOS device at every cross-

point [200]. While this design increases the size or overhead of the synaptic grid substantially, it may be useful for controlling non-ideal circuit effects, or it may be essential to realizing successful programming of the nano-synapse. One example of the second case is given in [201] which describes a 1T1R fabric used to emulate STDP with a metal-oxide ReRAM device. In [202], a three-dimensional 1T1R (called 1S1R in the paper) example is demonstrated using tantalum oxide memristive devices. While it has been argued that 1R designs are more susceptible to the sneak-path effect [203], the nature of the programming operation has a key effect on the severity or existence of this effect. In particular, if individual devices are read and write as in the context of memory, sneak paths are more of an issue; however, electrical simulations [204] as well as experimental works [67, 181, 182] demonstrate that neuromorphic-style programming schemes (updating an entire row or column of devices) are less susceptible to this effect. However, when scaling m/n to very high dimensions, the issue can arise again; as illustrated in [205], the non-linearity of the nanodevices then has a critical effect in determining the ultimate scalability of this 1R approach. Linn and collaborators have also proposed and experimentally realized a composite device called the complementary resistant switch (CRS) that naturally obtains very non-linear behavior [206], and verified its use in simple neuromorphic applications [207]. Another powerful engineering solution to this issues has also been provided in the form of specially engineered cell selectors or diodes, such as the recently realized FAST selector [208]. This solution is critical to realizing some pre-industrial realizations of 3D resistive-RAM ReRAM with extremely high accuracy and speed. For instance, Crossbar Inc. [24] highlights the importance of this 1D1R, or as they call it 1TnR, approach to accesses rows of devices simultaneously.

1.4.4.2 Random Topology

Unlike the ordered topology of a crossbar, the alternate approach employs random networks with an arbitrary number of neuronal nodes and synaptic edges/connections, resulting in unordered computational graphs. Examples of such architectures include the atomic switch network (ASN), self-assembled nanowire networks, and self-assembled forests of carbon-nanotubes. These sorts of systems are often fabricated using a combination of standard top-down patterning and bottom-up self-assembly, often using an array of 'seed' nanoparticles to sprout the growth of the network. Despite being feasibly realizable, they are presently difficult to characterize, understand, and control for computational tasks. Typically, a small number of input and output ports are designated, while a far larger number of ports is used to read information about internal states or dynamics. Work by Sillin has shown that self-assembled ASN networks operate with emergent criticality, a close-to-chaos computational regime [209]. The implementation of ASNs in reservoir computing schemes, first proposed in [210], has since been expanded into analysis of their computational capacity [211]. Recently, an ASN system was used in a demonstration of logic function memorization [212]. These sort of approaches are also extremely defect and fault-tolerant, as demonstrated in [213, 214].

1.4.5 The Neural Unit Framework: A scalable on-chip architecture

Much of the remainder of this thesis introduces various schemes for on-chip learning, whether they be experimental or theoretical. In this section, we build greatly upon the explanation given in Section 1.4.3.2 to demonstrate the foundational learning architecture/approach that will be used and referenced extensively within the following chapters. We first explain the design choices that led us to this framework, then describe its function.

1.4.5.1 Motivation and summary of key advantages

In general, we have chosen the crossbar as an organizing nanofabric for learning as it presently attracts the most attention for future memory designs, due to its scaling towards terabit-density synaptic arrays, and upcoming industrial realization. Specifically, we have built upon a past approach previously referred to as the Neural Logic Block (NLB) concept, which is demonstrably resilient to sneak paths, device variability, and defects due to the parallel programming and training style [204, 215]. Since nanodevices always suffer from some device variability and imperfections [216], finding an architecture which is intrinsically adaptive would be a major advantage. Indeed, our considered architecture has a natural ability to overcome the failings of individual devices at the systems level [217].

In this architecture, parallel rows of memristive devices holding trained weights (conductances) are each connected to a neuron and processing unit. The neural unit may be either built entirely from CMOS, or also integrate memristive devices in a hybrid neuron, depending on energy and density requirements. Each row (memristors + respective neuron) of the system learns in parallel, and can be considered as a rough analogy to a single dendritic branch. In addition, each individual neural learner is computationally equivalent to an adaline, a generalization of the perceptron, and thus learns with an on-chip friendly adaptation of the classic Widrow-Hoff algorithm, also called the 'delta rule' [119]. Building a nanofabric optimized to implement the delta rule also allows us to implement stochastic gradient descent in an entire crossbar, and naturally opens the doorway to more complex architectures as well.

To achieve such structures, individual learners can be chained together in a single crossbars to solve a problem in parallel. More elaborately, several crossbars of these learners could be connected to form a more expressive multi-layer feed forward neural network [218]. These approaches are illustrated in the top of Fig. 1.6, which shows how individual learners can be cascaded in a single or multiple crossbars. Since a crossbar of closely connected analog nanosynapses can be sequentially or simultaneously accessed in both programming mode, when applied voltage pulses are greater than thresholds, or inference mode, when applied pulses are below them, the energy efficiency of such structures should be very high since individual cells in the array do not need to be powered on/off. Lastly and very significantly, bottleneck computational operations in software neural networks, *e.g.* matrix-vector multiplications, can be naturally performed on-chip in the crossbar configuration [219, 220]. This makes them a natu-

ral building block for on-chip neural networks.

Due these favorable electrical and parallelization features, this design has high forward compability with the 1TnR (1 row of resistive device per selector or diode) ReRAM systems earlier discussed. All branches/learners in a given layer/crossbar are connected to a finite state machine (FSM) which provides programming and input learning pulses [221]. If combined with the ultra-high density memristor/CMOS neuron design, this proposal could even increase the energy efficiency of proposed 1TnR designs by reducing the number of control transistors typically needed to implement parallel programming in a large synaptic array.

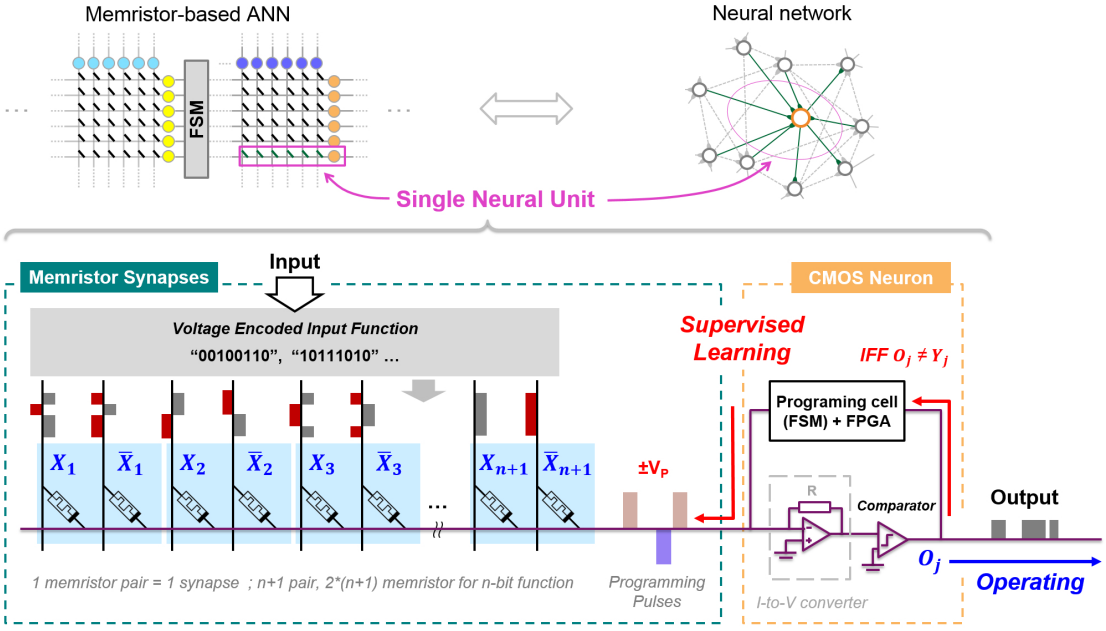


Figure 1.6: Schematic representation of the neuromorphic learning system.

1.4.5.2 Operation of canonical circuit

Our canonical adaline circuit encodes weights using signed-synaptic pairs of memristive devices with modifiable conductivity; $n+1$ pairs, or $2n+2$ memristive devices, are required to successfully map a function with n inputs. Each of the $n+1$ inputs requires a negative and positive wire to separate states for that case, and negative and positive bias lines configure the entire line. In response to a set of input voltages, the sum of all conductances is proportional to current on the common post-synaptic line for that particular branch. After said current is converted to voltage and digitized, the sign (+,-) of the output is compared to the sign of desired function (teacher signal). If they are different, a programming cell applies an appropriate correction pulse. A depiction of this is also visible in Fig. 1.6, bottom system schematic.

The Widrow-Hoff (WH) algorithm or delta rule, which solves the least-mean squares problem by stochastic gradient descent, provides the engine by which weights are successfully trained

to map any linear function perfectly, and more complex ones imperfectly [118, 222]. This system was already conceptually introduced as well in Section 1.3.5. Relative to this design, our on-chip friendly version of WH is simplified in one key way: the difference between teacher (expected) and actual output signal is passed through a binary filter ($sign$), such that only three cases exist: same sign (0); different sign, low error (-1); different sign, high error (1).

Online learning with binary WH is implemented step-by-step; at each step (epoch), binary difference between expected (O_j) and actual output (T_j) for the neuron j is evaluated and, if an an error case exists, the appropriate adjustment is made. The first error case, low error/-1, exists when $O_j < 0, T_j > 0$; the second, high error/1, when $O_j > 0, Y_j < 0$. In the former case, WH *increases* the value of pairs connected to positive lines, and decreases the ones connected to negative lines so that output rises. In the latter case, WH does exactly the opposite: *increases* the weight of all pairs connected to negative lines, and decreases pair weights on the positive lines so that output falls. In order to achieve this, error correcting pulses V_{p+}, V_{p-} on the post-synaptic line program every device according to the voltage from the pre-synaptic line. In particular, voltage difference across a given device (EDP) automatically determines whether conductivity increases, decreases, or remains constant at that particular active (error-correcting) moment. The four active ($T_j \neq O_j$) steps that implement on-chip binary WH are demonstrated in Table 1.3 below, considering every possible combination of sign on the input X_i, sign of the output O_j, and sign of target function T_j. In particular, Steps 2,4 (S2,S4) correct the low error/-11 case, and Steps 1,3 (S1,S3) improve the high error/1 case.

X_i	Y_j	O_j	ΔW_{ij}	Step
-1	-1	1	1	S1
-1	1	-1	-1	S2
1	-1	1	-1	S3
1	1	-1	1	S4

Table 1.3: Active programming steps that solve the delta rule.

1.4.5.3 Implementation with a nanodevice

In the following Chapter, we consider the implementation of this abstract learning system and its rules in the context of an actual emerging nanodevice. In this context, adaptations to the standard scheme are made to account for characteristic device behavior, and resilience of the framework to imperfect physical phenomenon is assessed.

Chapter 2

Computing with organic nanodevices: theoretical and experimental work

Imagination is more important than knowledge.

Albert EINSTEIN

"THIS CHAPTER *builds upon the neuromorphic inspiration already introduced to show, using phenomenological device models as well as physical memristive devices, how a practical on-chip learning system may be implemented electronically. Characterization, modeling, and experimental learning demonstration of an organic nanosynapse are all highlighted in this chapter. A broad discussion is ultimately fostered in considering the merits of each of these approaches and the ultimate prospects for building neuro-computers with highly analog organic memristors.* "

THIS CHAPTER focuses in-depth upon a single memristive device, a complex polymeric redox filamentary nanodevice, and shows its physical properties, behavior, and performance in theoretical, electrically simulated, and experimentally realized learning systems. There are five sections that explore the following topics respectively:

1. General motivations are given for exploring this novel device (2.1)
2. A materials physics, electrical, and behavioral introduction to the device is provided. (2.2)
3. Proof-of-concept learning of the basic neural crossbar scheme and automatic error-programming for local learning is demonstrated with electrical simulations using a simple functional compact model (2.3)
4. More complex aspects of device behavior are analysed with the introduction of a more complex and physics-oriented compact model. On this basis, more complex electrical simulations are introduced (2.4)
5. Experimental learning of a single learning system is implemented, and key results presented, with the emphasis on non-ideal effects that were different than electrical simulations. (2.5)

2.1 Motivation for studying novel organic nanodevices

Memristive devices made with organic materials offer unique advantages: high retention and strong data storage properties [223], low material costs by using renewable or cheap chemical materials [224], scalable fabrication via roll-to-roll imprint lithography [225], compatibility with flexible substrates [226], and possibility of three-dimensionsal array integration [227]. In addition, fabrication allows for rich structure flexibility through chemical synthesis, and room temperature processing ability [228]. These properties pave the way towards integration with embedded sensors, bio-medical devices or wearables, and other internet of things (IoT) applications. One particularly appealing vision is the combination of organic nanosynapses with organic transistor devices into all-organic learning systems [229] in cheap or even disposable electronic devices for ambient computing.

In addition to their marketability for building future neuro-synaptic technology, from the material physics perspective organic memristors are particularly fascinating. Several other species exist and have been listed in the literature depending on the considered polymer and/or the materials used for the electrodes; most of them are presented in [54]. Their switching bears similarity to other filamentary ReRAM, but with an added layer of complexity based on the electrochemical behavior of the active switching material. Generically, organic memristor devices rely upon conformational changes of a thin polymer layer/film between a highly conductive

(oxidized) state and a highly resistive (reduced) one [230]. In some cases this is a well known redox reaction, while in other cases it may be a more complex ensemble of effects that resemble switching in inorganic memristors, *e.g.* reversible filamentary conduction based on anion- or cation- migration [55]. Lastly, some switching mechanisms are specific to organic memristive devices, such as chemical conformational effects. In addition, sample preparation and morphology, electrode materials, interfacial properties, testing methods, and circuit environment all contribute towards the final electrical characteristics of a given set of memristor devices [231, 232]. Within this complex context, we aim not only to adequately characterize our novel new device, but to show its practical applications.

Our work is not the first to attempt to integrate some variety of organic memristive device into a hardware learning system. For instance, [233] integrated polyaniline polymeric devices previously characterized in [234] into an ANN set-up. However, the programming durations for this set-up were far too slow (30s per programming pulse) for IoT applications that require online learning. This trade-off is not unique, as the cost of slower programming in highly analog memristive synapses relative to inorganic memristive devices or binary organic memory devices is mentioned often in the literature [235, 236]. In this sense, the fact that the TBFe device we study here is capable of learning at speeds relevant for modern applications (100 μs per programming pulse) is a significant asset. Next, we introduce the physical properties of our device that make this sort of operation possible.

2.2 Physical properties of TBFe Memristor

Through a partnership with V. Derycke's group at CEA-IRAMIS as a part of the ANR-MOOREA project [237], we have worked on the characterization as well as experimental demonstrations of a novel form of electro-grafted organic memristive device that has several very favorable properties for future applications. In this section, we introduce the physical parameters of this exciting new device.

The organic memristive device has as its active layer a polymeric film electro-grafted in between metal contacts; an applied current implements memristive behavior as conductive filaments are created or destroyed between the two electrodes above or below given critical thresholds. The electro-chemistry of the polymer used as well as the metal type and work function of the electrodes determines the conductance evolution graph. The switching media is a thin-film polymer of tris-bipyridine $Fe(bpy)_3^{2+}$ iron complexes (TBFe), where memory effects emerge as a dynamic redox system [238, 239]. Chemically, each molecular complex ${Fe(bpy)_3}^{2+}$ contains three bi-pyridine ligands surrounding a central iron core with three diazonium functions, which allow for covalent bindings between molecular complexes, and between the complexes and the electrodes.

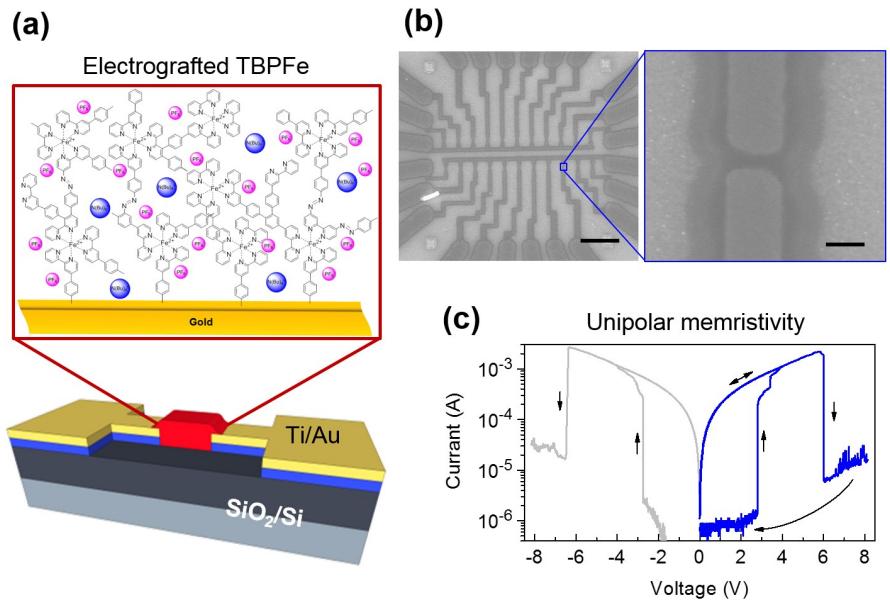


Figure 2.1: (a) Schematic representation of the metal/organic/metal memristor and the organic-composing active layer. (b) SEM image of the actual devices. Scale bar in the left and right images represent 20 μm and 200 nm, respectively. (c) Electrical characteristics of the memristor under voltage sweeps.

Individual devices consist of metal/organic/metal junctions fabricated on an oxidized silicon wafers (SiO_2/Si substrate) . A series of Ti/Au electrodes are then fabricated by e-beam lithography, evaporation and lift-off on top of the substrate to yield gaps between the electrodes ($\approx 30nm$). A thin (height $\approx 10nm$) organic film of covalently bounded TBFe complexes is then formed by electrografting in between the two gold metal electrodes; this forms an electrochemical cell [239]. Next, the three diazonium functions on the iron complexes are electrochemically reduced via cyclic voltammetry (CV) technique. The radicals formed covalently bond the molecules to the electrodes and to each other to form a compact and robust thin-film [240]. Critically, the redox properties of iron complexes inside the electrografted film are preserved during this process. Once formed, the robust thin film consists of about 6-7 molecules in the vertical direction and 20 molecules along the gap direction. TBFe cations are counterbalanced by PF_{6-} anions. Further Details on material synthesis and the experimental electro-grafting process are presented in Sections A.1.1 and A.1.2, respectively.

Regarding fabrication environment, electrochemical deposition is fast and takes place at room temperature in a mild chemical environment. It enables the localization of the functionalization and the fine control of the film parameter [241, 242], and also allows the local deposition of different compounds on the same chip. It is thus fully compliant with preexisting structures or devices on the chip, allowing heterogeneous co-integration of synapses and neurons in future designs. In addition, our device possesses $G_{\text{Max}}/G_{\text{Min}}$ ratio above 10^3 and endurance above 2000 SET/RESET cycles, as visible in Fig. A.3.

A schematic representation of the device and an image of a series of organic memristive devices under scanning electron microscope (SEM) are visible in Fig. 2.1a,b respectively. If unencapsulated, the present devices behave as a unipolar conductive-filament memristor only in vacuum (10^{-2} Torr). In future designs, encapsulation should allow for memristive behavior at room temperature/pressure. The characteristics of this device and its immunity to the scaling of surface junction imply a dynamic filamentary behavior at the active region. Although a planar structure was employed in this work, the vertical structure of the device shows comparable switching performances as horizontal ones. This demonstrates the possibility for integration in high density crossbar arrays. Further information and an additional experimental realization of this vertical approach is presented in Section, A.3.

2.2.1 Electrical Regimes

Our device possesses two electrical thresholds. If voltages larger than the first (V_{th1}) are applied to the device, its conductance increases non-linearly (SET mode); if voltages larger than the second (V_{th2}) are applied, its conductance decreases non-linearly. Considering the non-volatile regime, which corresponds to all applied voltages below the first electrical threshold, there are three operating modes. These thresholds and modes are all visible in the I/V curve in Figure 2.1 (c), where the arrows demonstrate the directionality of the voltage sweep producing the curve. As visible in this curve, conductance evolution is symmetrical (the memristor is unipolar): negative and positive bias is applied in the given ranges have the same effect. Another positive asset of our organic memristive device is that it *does not* require a compliance current to form an initial filament and thus allow 'Set' mode operation, as many other unipolar [243] as well as bipolar [244] non-volatile memory devices require. This is a decisive asset from a circuit point of view; since the initial forming step can require a high current, device initialization can be difficult or damaging inside of a dense crossbar structure with this constraint.

Fig. 2.2a shows the conductivity evolution with pulses of increasing amplitude. Series of 100μs long programming pulses are applied by increasing 0.25V every 15 pulses. Conductance evolution is monitored at 0.5V between each pulse. The actual applied waveform is shown in the inset of Fig. 2.2a. A measurement cycle begins at 2V and ends when the device returns to a low conductance state. The black curve shows one representative cycle of the total ~2300 cycles that were applied (grey curves). These measurements show the possibility of reaching many intermediate levels with short pulses during the SET process (between 3 and 5V), while a stable state is more difficult to obtain during RESET, which dramatically decreases conductivity above 5V. Asymmetry between SET and RESET modes is typical of dynamic filamentary behavior, and constitutes a general limitation learning schemes must address [170].

The top panel of Fig. 2.2b shows the change of conductivity (ΔG) as a function of pulse amplitude, based on the statistics of the former measurement. ΔG is defined as the difference between the device conductivity before and after applying the pulse. The red curve shows the average ΔG for each pulse amplitude. The two thresholds, V_{t1} and V_{t2} distinguish the SET and

RESET region and are used for the learning algorithm. It should be noted that, in practice, most pulses induce very little change in memristor conductivity. The lower panel of Fig. 2.2b shows the SET/RESET event probability with respect to the pulse amplitude, where a SET/RESET event is counted when $\Delta G/G_0 > 40\%$. These device characteristics are essential to optimize SET/RESET pulses amplitudes for the learning algorithm, and for the memristor to properly display synaptic function in a neuromorphic system.

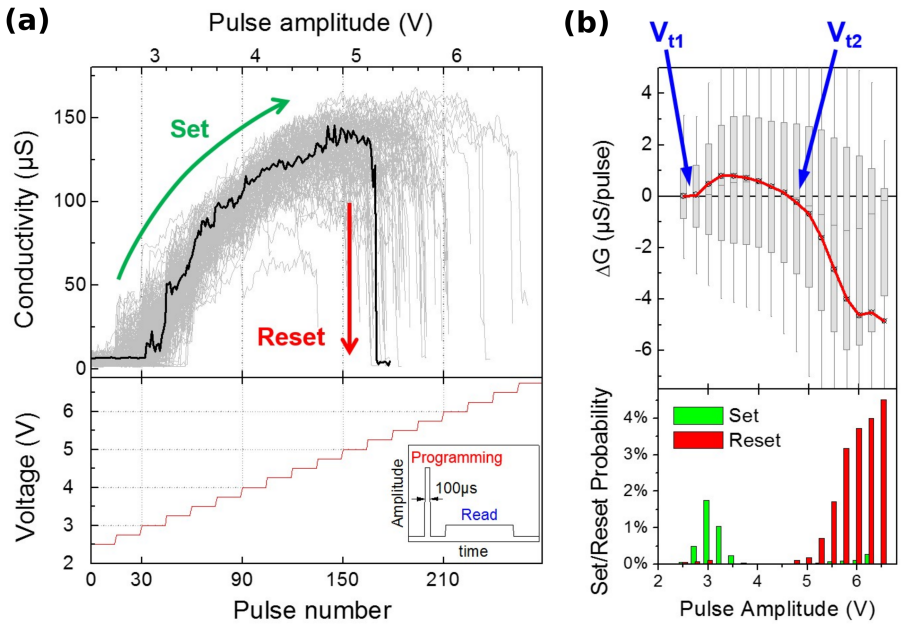


Figure 2.2: (a) Top panel: Conductivity (G) evolution of the device under pulses with increasing amplitude. Gray traces show all transitions and one characteristic transition (black) is highlighted. Bottom panel: amplitude of each pulse. Inset: representation of the applied waveform. (b) Top panel: Statistics of conductivity change (ΔG) versus pulse amplitudes. The gray boxes show the 25%-75% probability and the whiskers are 10%-90%; bottom panel: SET/RESET event ($\Delta G/G_0 > 40\%$) probability with respect to the pulse amplitude.

2.2.2 Origins of dynamic redox behvior

In the TBFe thin film layer, dynamic filamentary behavior results due to conductive paths being formed or destroyed. A schematic of the individual TBFe complexes surrounded by negative anions and their connection through both "azo" bridges and carbon bonds is visible in Figure 2.3(b); Fig. 2.3(c) shows a conceptual depiction of how filaments (green) may grow and evolve within this environment over time. Our current hypothesis is that the change in conductivity within filaments arises from reversible charge separation between the iron core, the ligands and the azo-bridges. At the bottom most image roughly corresponding to G_{off}, some TBFe complexes connect but none provide a clear path between the electrodes; in the intermediate states, the domains develop; finally, in the top-most state corresponding to G_{on}, many

conductive filaments exist between the two electrodes. A high voltage reversibly disrupts these conductive filaments.

A more detailed switching mechanism hypothesis based on intramolecular charge transfers in analogy with percolation models was made in [239]. In particular, our device's metastable redox behavior most likely is related to two different key molecular bonds forming and breaking: conventional C-C bonds of low conductivity and azo-bridge bonds that can switch between aromatic and quinoid conformations. Azo-bridges bonds are more conductive in quinoid form than in aromatic one. At $V > V_{\text{th1}}$, an intramolecular charge transfer associated with the oxidation of the iron ion ($Fe^{2+} \gg Fe^{3+}$) occurs and is stabilized by a conformational change of the bond which is turned to its more conductive form. This central redox operation is visible in Figure 2.3(a). Above the second threshold V_{th2}, electrons injected from the metal contact induce reverse switching. Depending on the proportion of molecular bonds stabilized in the conductive state, the film presents either complete or incomplete conductive paths. This rich behavior allows for stabilization at varying levels of conductivity. Further discussion on the nature of the filamentary behavior is in Section A.3.1.

2.2.3 Dimensions and Scaling

Device tunability for different applications exists as a function of the selected molecular compounds, grafted parameters and device geometry. With regards to the latter case which is most directly modifiable in the fabrication cycle, current-voltage plots are shown for the initial (forming) step in Figure 2.4 (b). As visible, shortening the inter-electrode distance L has a discernible effect on the device's electrical behavior; downward scaling yields reduced required WRITE and ERASE voltages (by lowering V_{th1}, V_{th2}). This is a favorable trend which could be exploited in the future to engineer smaller organic nanodevices that are even more easily co-integrated with standard CMOS technologies.

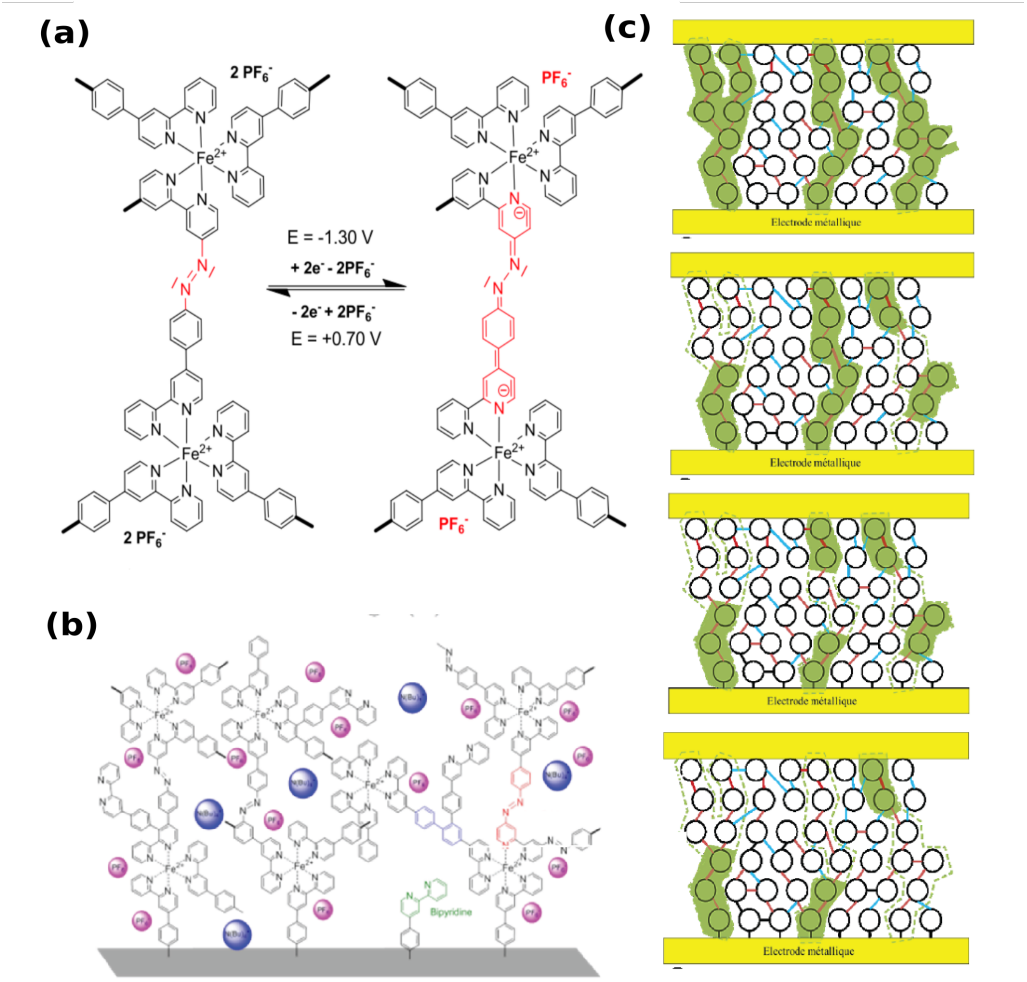


Figure 2.3: (A) Two configurations of the central Iron core TBFe molecule are highlighted, with conformational change (redox change) noted. B) Schematic of chained TBFe molecules into a compact thin-film structure C) Filamentary development within the thin-film as current is applied between the two electrodes; conductive areas shaded in green. Source: [238].

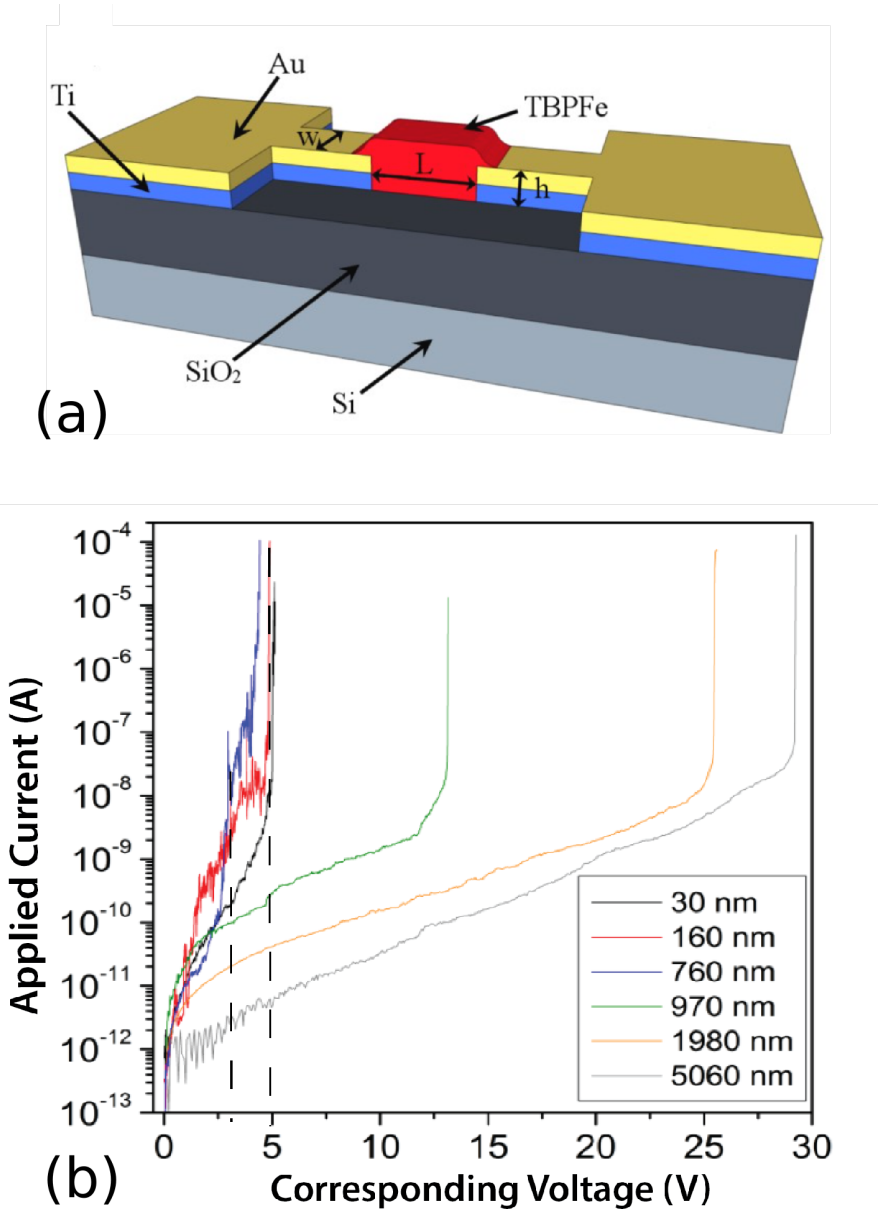


Figure 2.4: (a)3D rendering of the planar device, where red is the electrografted active layer, with key dimensions (L, inter-electrode instance, h, electrode height, and w electrode width) noted. (b) Current as a function of voltage at the forming stage, measured for different device size (inter-electrode distance L). The two dotted lines show the first and second threshold for the hardware device. Both adapted from [238]

2.3 On-chip learning with organic memristive devices

2.3.1 Simple Functional Model

In a simple bipolar memristor model, conductance does not vary when the device is exposed to a given voltage value in between the thresholds (below the positive threshold V_{th} and yet above $-V_{th}$); else, it increments at a voltage above the positive threshold or decrements at a voltage below the negative threshold. A unipolar two-threshold device such as the organic TBFe species evolves its conductance by a notably different scheme. When a given voltage value applied at time t: $V(t) = V_{\mathrm{M}}$ has an absolute value greater than first threshold V_{th1} and smaller than the second threshold V_{th2} an increment to conductance occurs (WRITE mode); when its absolute value is greater than V_{th2} a decrement occurs (ERASE); and when its absolute value is less than V_{th1}, no conductance change occurs (READ). The simplest model to express these ideas is:

$$\frac{dG}{dt} = \begin{cases} V_{th2} > |V_M| > V_{th1} & \alpha(|V_M - V_{th1}|) \\ |V_M| > V_{th2} & -\beta(|V_M - V_{th2}) \\ |V_M| < V_{th1} & 0. \end{cases} \tag{2.1}$$

Constants α and β were set based on early experimental results with the organic memristor. Notably, $\alpha < \beta$, as noted in experimental results. Integrating over a small time window, $10^{-8}s$, is equivalent to a positive conductance change of $\Delta G = 7 \times 10^{-8}S$ in the write/SET mode, and over an equivalent integration window, $\Delta G = 1.2 \times 10^{-7}S$ for RESET, which is nearly twice as powerful. In addition, as noted in Eqn. 2.1, voltage values further from the threshold in both regimes produce large ΔG values. While a maximal conductance increment voltage value is given at the absolute voltage value just below $|V_{th2}|$, the maximum value for decrement is given at an absolute voltage value just about 0.5V above $|V_{th2}|$, and saturates at this maximum rather than linearly scaling at even higher voltages. However, in our case, these values are not modulated, yielding to typical ΔG values for both SET and RESET the learning process. Note this is a physically unrealistic assumption, but useful as an early benchmark of possible performance. Lastly, early experimental results also suggested (G_{off}=150nS, G_{on}= 100μS). This simple uniform model for unipolar organic memristor was implemented in Verilog-A, an all-analog, continuous time sub-set of the Verilog AMS hardware language- and integrated into electrical simulations in the context of the Cadence Spectre electrical simulation environment. It is used for all simulations of our unipolar organic memristors in the following section.

2.3.2 Mapping Learning Scheme to Organic Device

First, we demonstrate a mapping of the generic memristive learning scheme using binary Widrow-Hoff (WH), or delta rule, learning as shown in Section 1.4.5 to our particular device. Once again, the objective is to implement the following weight update at each synapse connected to pre-

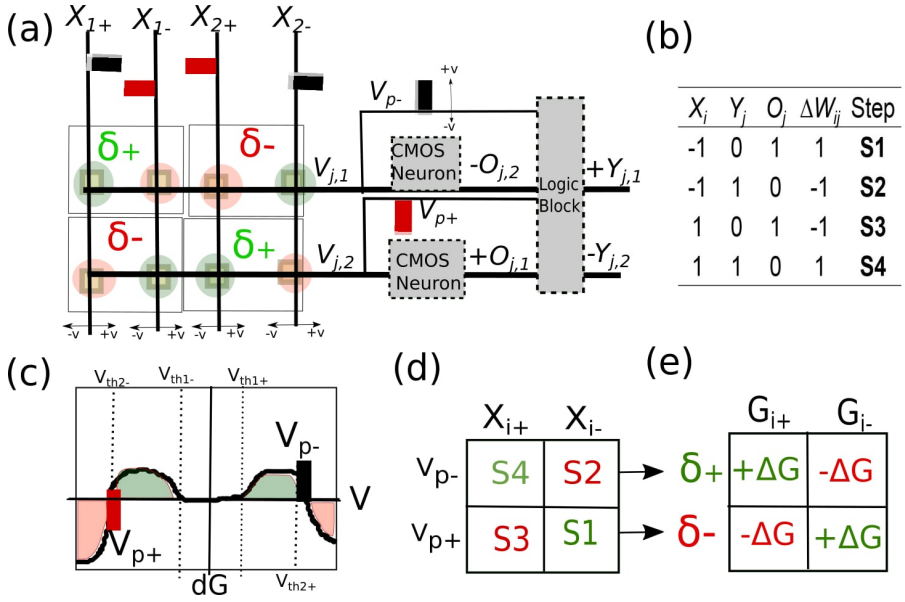


Figure 2.5: (a) is a conceptual schematic of a crossbar with two parallel neural learning systems, where the top row is in the LH error case ($O_{j,1} < 0, Y_{j,1} > 0$) and the bottom in the HL error case ($O_{j,1} > 0, Y_{j,1} < 0$). (b) Shows the truth table for implementing binary WH learning. (c) shows the conductance evolution of the device, in particular δG as a function of a voltage V, with the two utilized conditional programming pulses, V_{p-} and V_{p+} noted. (d) shows the implementation of the error-correcting steps shown in (b) by these programming pulses, while (e) shows implemented conductance changes ($\delta-$ is S3,S2; $\delta+$ is S4,S2). Consistent color coding between (a), (d), and (e) shows the implementation of delta rule on different scales.

synaptic neuron i and post-synaptic neuron j:

$$\Delta W_{ij} = \alpha sign(X_i) sign(T_j - O_j) \tag{2.2}$$

where α is a constant and in practice relates to the size of the programming pulse, and W_{ij} is physically represented as a pair of devices to realize a full range of weights ($W_{ij} = G_{ij+} - G_{ij-}$). These updates minimize prediction error on a given task by implementing negative or positive increments towards the correct configuration in successive steps. Physically, this requires us to lower the aggregate conductance of every pair of devices whose output is too high, and inversely physically increase the conductance of every pair of devices which is too low.

From the on-chip perspective, binary WH is implemented by comparing sign of output (O_j) and expected (Y_j). Therefore, there are two relevant error cases: when $O_j < 0, Y_j > 0$ (Low/High (LH), or -1), and when $O_j > 0, Y_j < 0$ (High/Low (HL), or 1). In the former case, WH increases the value of all pairs such that output rises to meet expected. In the latter case, WH decreases the weight of all pairs such that output falls to meet expected. Error-correcting pulses V_{p+}, V_{p-}, are back-propagated from an on-chip learning cell along the post-synaptic line of our system, as depicted in Fig. 2.5(a). Crucially, these programming pulses have *differential impact* de-

pending on whether the input line has a positive or negative voltage polarity. In turn, the combination of pre-synaptic and post-synaptic voltage potentials at each device determines whether conductivity increases, decreases, or remains constant at that particular active (error-correcting) cycle of the given epoch. Since input can be high or low, two error cases become four active steps that implement binary WH completely: Steps 2,4 (S2,S4) improve LH error, and Steps 1,3 (S1,S3) improve HL error (Fig. 2.5(b)).

Appropriate programming pulses V_{p+}, V_{p-}, are determined by the conductance evolution of the device [245]. Our device possesses two thresholds and is unipolar in conductance evolution, which offers a unique advantage. As visible in Fig. 2.5(c), programming pulses applied at the second thresholds ($V_{p+} = V_{t2-}, V_{p-} = V_{t2+}$ allow the system to correct all four error cases (Fig. 2.5(d)) when the voltage polarity is opposite of the sign of the error case. This simultaneous programming step can also be seen in the crossbar schematic (Fig. 2.5(a)) where the $\delta-$, $\delta+$ improvements on each pair that directly satisfy binary WH are noted (Fig. 2.5(e)).

This is notably more economical than the on-chip binary WH rule for bipolar devices, which not only requires four pulses but maintains separate lines for the momentary inversion of all inputs to program successfully. This requirement has to do with the conductance asymmetry evolution of these devices, as described in [245]. While these energy savings may seem small in this context of a single learning operation, for very larger functions presented over hundreds of epochs, this could represent significant savings. This highlights the favorable properties of our symmetric devices, and motivated the development of a hybrid memristor/CMOS neuron design for even greater energy and density benefits.

2.3.3 Building electrical neural learners

In this section, we extend the conceptual neural model just described to a more concrete electrical implementation. There are three components to this electrical system, as depicted in Figure 2.6:

- The dense crossbar of memristive devices which realizes adaptive weights.
- Simple neuron thresholding functions (realized in CMOS)
- Complex neural logic or learning cell (realized in CMOS and with memristive devices separate of the crossbar)

First, regarding the crossbar of adaptive weights, if we want an input with n inputs to be recognized, $2n+2$ physical wires and physical devices are required to achieve this. In particular, each input stream X_i, requires two nanowires (X_{i+}, X_{i-}) at whose intersection with row j, two memristors (M_{ij+}, M_{ij-}) encode a unique synaptic weight pair as a difference function ($G_{ij+} - G_{ij-}$). This approach is visible in Figure 2.6 where the blue, red and green pairs of devices encode adaptive weights that relate Input 2, Input 1, and Bias respectively to the common

teaching line. Second, for every neural learner we obtain a post-synaptic potential V_j automatically by the combination of Ohm's and Kirchoff's Laws. Once inverted at a simple neuron (CMOS inverter) with ground as the threshold, we obtain the final output state O_j which is high (+1) or low (-1):

$$V_{jb} = O_j = sign(V_j) \tag{2.3}$$

The output of this inverter, O_j is subsequently fed to a set of transistor switches through which it is compared to the target output, Y_j. Third, after error cases are obtained for every neuron, configuration is required to appropriately route error-correcting pulses to each neural learner. In practice, these configurations are achieved automatically using complex logic within the learning cell, as will be described in the following section. Ultimately, once both the neurons and learning cells have completed their activities following each presentation, application of programming pulses V_{p+} and V_{p-} occurs, during which correction pulses are passed from the learning cell back along post-synaptic line, simultaneously correcting all existing error cases in the crossbar.

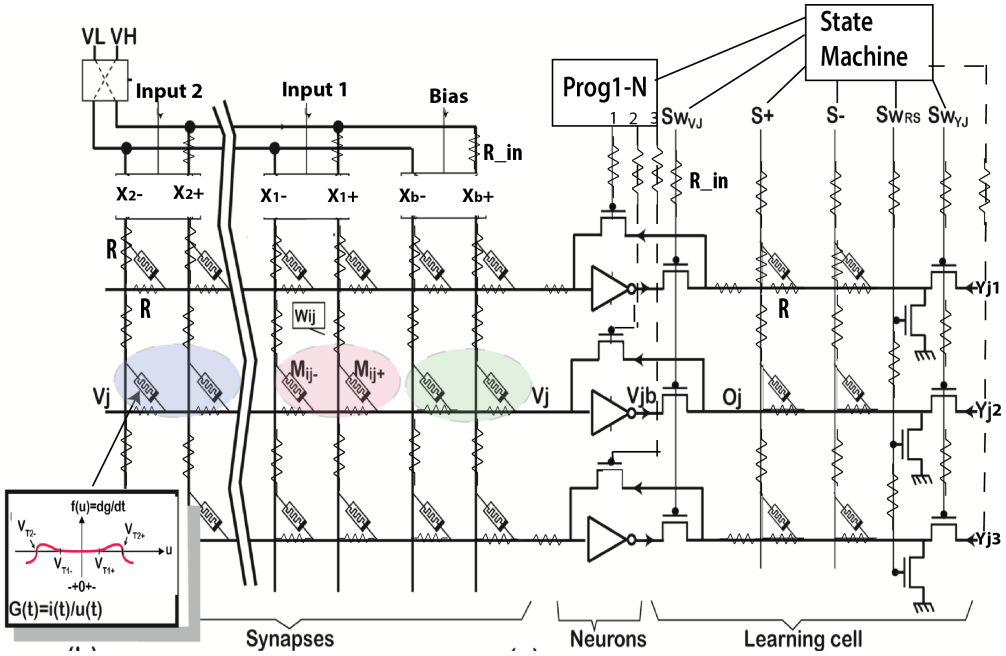


Figure 2.6: Schematic for an all-unipolar neural crossbar setup (all constituent memristors follow conductance evolution graph in inset). Each row takes a separate programming pulse per cycle so SW_{prg} is now SW_{PrgJ} for each row. No input polarity I_p. Parasitic resistances are materialized by the R and R_{in} variables. Adapted from [245]

2.3.3.1 Working principle of hybrid learning cell

Within the learning cell, a pair of memristive devices separate from the main synaptic adaptive grid/crossbar encodes a logic function through its states. In particular, as discussed in [246], the pair of memristive devices implements an IMPLY logic gate, and this conditional logic is

used to achieve intelligent programming.

In particular, we automatically program neural learners by connecting the input of the dual memristive devices to control two signal lines, S_+ and S_-, which are sent from a nearby state machine that controls the overall learning process. The signal lines S_+ and S_- directly feed into a set of memristive 'latches' $A_1..A_j$, $B_1..B_j$, or the positive and negative latch for each learner system. Meanwhile, each set of memristive latches is of course connected to its own appropriate output line O_j. This is all visible within Figure 2.6.

In the standard latch programming scheme, demonstrated in Fig. 2.7(a), the two signal lines are connected to two memristor latches in opposite orientation (*e.g.*, one has its positive pole facing the common line, while the other has its negative pole facing the common line). When the polarity of the two memristor devices along the signal lines are reversed, the two devices implement exactly opposite responses to applied voltage V on the line. This scheme requires four control transistors, which are controlled by gate voltages SW_{Vj}, SW_{PR}, SW_{RS} and SW_{Yj}, respectively. These transistors connect or disconnect signal lines from each other at various moments in the learning cycle. This process is visualized in the bottom panel of Fig. 2.7 (a). There are two overall steps: configuration phase, where the latches are prepared to correctly route, and programming stage, where pulses are actually applied.

In the configuration step, the output or common line O_j is first opened; this is an 'unconditional open' that RESETs the state of all devices. Second, the common line facing both S_+ and S_- is connected to V_{jb}, which implements a conditional close at the latch's thresholds $-V_{th}$ and $+V_{th}$, respectively. This means that A_j will RESET (close) and B_j will SET (open) conditional on the value of $V_j b$, as noted in Fig. 2.7 (a). Lastly, the common line is connected to the target Y_j, implementing the opposite operation (conditional open). Precisely, A_j will SET (open) and B_j will RESET (close) depending on the value of the expected output. Taken together, these last two steps program the anti-parallel memristive devices for all neural learners to exactly correspond to the error case it possesses ($V_j b$ and Y_j cases may be different at each line/learner).

In the programming step, the V_{p-} programming pulse is first sent through line S_+, reaching all learners where A_j is open, and automatically correcting will the LH/-1 error case in all of them (S2,S4 steps implemented at every pair). Finally, V_{p+} programming pulse is sent through line S_-, reaching all learners where B_j is open, and automatically correcting the HL/1 error case in all of them (S1,S3 steps implemented at every pair) [221].

2.3.3.2 Unipolar memristive latch scheme

We have built upon the above generic scheme by proposing to directly integrate our unipolar devices into the learner cell as well as the crossbar (*e.g.*, an all-organic scheme for on-chip adaptation and logic). We first note that the proper logic voltage level for programming pulses now sits at the second threshold (V_{th2} or $-V_{th2}$), since slight changes in either direction can induce an increment or decrement. The polarity of the pulse must follow the sign of the expected output for the row or function, Y_j , with the condition that $V_{p+} = -V_{th2}$ and $V_{p-} = +V_{th2}$ (since

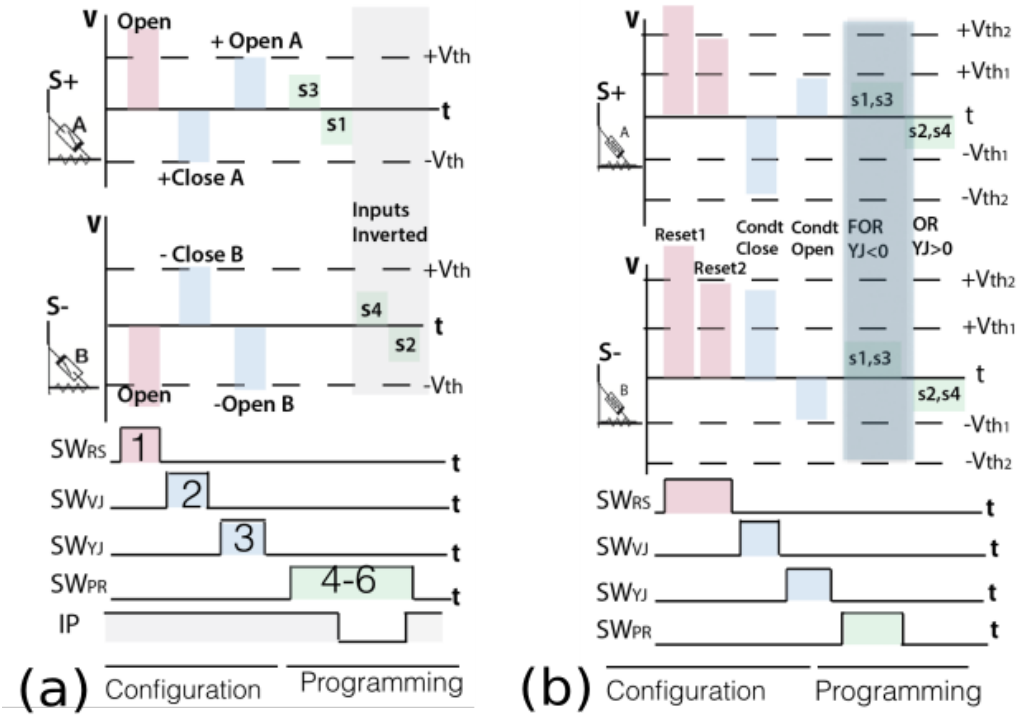


Figure 2.7: A): One complete configuration and programming phase through time t is shown for a pair of generic memristor logic devices which implement the automatic programming within the learning cell. B): Our adapted all-organic scheme is demonstrated. More details on both of these schemes are provided in the text.

conductance drops along the second threshold) [245]. Given this, a single pulse fed simultaneously to signal lines S_+ and S_- can then implement learning with only one programming pulse per cycle. If $Y_j > 0$, the programming pulse V_{p-} satisfies both S2 decrementing conductance and S4 incrementing it depending on whether the input line is low or high respectively; if $Y_j < 0$, the programming pulse V_{p+} implements S3 decrementing or S1 incrementing conductance when input is high or low respectively.

This all-organic scheme for implementing both the synaptic grid and the memristive neuron design with the two-threshold unipolar device is visible in Figure 2.7 (b). Note first that in this case anti-parallelism is no longer required- the devices are arranged exactly like in the normal analog array. As in the other cases a large pulse is needed to unconditionally open the latch; here it is the value just below $V_{th2} = 4.8V$. Moreover, due to the analog nature of the latches, it was found that a large negative pulse $V = 5.5V$ was needed just before this to properly reset states for each cycle due to gradually climbing conductance levels. Note that, to realize this scheme successfully, different threshold levels are needed which could be realized by different device scaling parameters. Values of V_{th1}=1.2V and V_{th2}=2.5V were chosen for the analog array memristor; a larger device is used in the learning cell, and its threshold values $V_{\text{th1}} = 3.3V$, $V_{\text{th2}} = 4.8V$, closely correspond to the experimental values from Figure 2.1 (c).

While conditionally closing and conditionally opening the unipolar device latches is more or less synonymous in polarity with the earlier approach, it requires the use of pulses at both thresholds in order to function properly. Since the programming pulse for the row devices must also sit at their own second threshold voltage, the first threshold $|V_{\text{th1-Latch}}|$ of the latch must be greater than $|V_{\text{th2-Array}}|$ so as to keep the analog latch safely in its configured state while these pulses are applied at the end of each cycle. This requirement has motivated the use of two differently sized organic devices, a possibility offered by organic memristors.

Unlike the bipolar case, the dark blue or white programming moments are not both required; only one pulse is sent for each error-type (V_{p+}, V_{p-}). Thus, in a multi-neural unit system where both types of errors exist, two pulses applied through the programming lines will solve all error cases in the crossbar. In this case, the gate voltage level for each of the programming transistors SW_{PrgJ} ensures that each row only gets the pulse it needs.

2.3.4 Simulations of memristor/CMOS hybrid neuron system

The objective of this section is to provide a proof-of-concept by integrating the simple functional memristive analog model, the electrical crossbar set-up (neural learner scheme), and automatic programming in the learning cell in order to concretely demonstrate high-density on-chip learning. We depict successful learning of simple logic functions before discussing the impact of a key electrical system-level defect.

2.3.4.1 Electrical Simulations: Logic functions

As our first demonstration, we chose to wire a crossbar of memristive devices such that each row (neuron) can learn a linearly separable function given an arbitrary number of logic inputs (bits). Again, if we want a logic function of n bit or inputs to be learned/recognized, $2n+2$ physical wires and physical devices are required to achieve this- double for each bit and an extra two devices for the bias lines. Using delta rule to memorize a temporal sequence (here logic function) can be achieved by presenting all possible cases of the function (truth table) sequentially and updating weights after each one has been given. Given a logic function of depth n bits to be learned, 2^n cycles per epoch are required to map a full presentation of the truth table to the nanodevices. In the following simulations, we are only learning the simplest (2 bit) logic functions, hence there are 4 cycles per epoch. In our following simulations, we set 1 epoch as $4\mu s$. Epochs repeat until the function is learned.

In this section, we verify the successful integration of the proposed analog device and the on-chip neural learning scheme using the Cadence Spectre environment to perform electrical simulations. All nanosynapse models in the synaptic grid or neuron used our own Verilog-A models, while the all CMOS elements were standard simulated using a commercial 45nm low power design kit. Transient simulations, visible as Figure 2.8 reveal that all considered automatic programming schemes learn successfully. Learning takes several epochs (presentations

of the target) to complete; it is finished around $19\mu s$, or just before the end of the entire learning period in both cases. For both hybrid and all-organic models it was seen that final conductance values stay low overall, evolving from 150ns to somewhere in between $5\mu S - 10\mu S$, one to two orders of magnitude change but nowhere near the $G_{\max}$ value. This is mostly due to the fact that the simple conductance evolution models has specified a higher β than α value, so conductance values always stay relatively low. Nevertheless, the fact that the model can still learn with a generic asymmetry case demonstrates the power of the algorithm to automatically adapt.

2.3.4.2 Electrical Simulations: Resilience to Parasitic Resistances

Lastly, we analyzed how parasitic resistances might degrade learning performances within our concrete electrical proposal for on-chip neural learners. Since metallic nanowire resistances can vary within orders of magnitude depending on diameter, length, grain boundaries, scattering, and contact resistance issues, resilience to non-negligible wire resistance is a critical topic in synaptic array design. In this section, we evaluated the ability of both memristor neuron designs- the all-organic and standard design- to properly learn 8 functions simultaneously over a wide variety of parasitic line resistances.The 8 functions chosen (in the left of Panel A, Figure 2.9) are the 8 non-trivial and logically separable options amongst 2-bit functions. In order to simulate this, resistors were placed along all connections in the simulated neural crossbar system, as visible already in Fig. 2.6.

At lower wire resistances (for $R < 5k\Omega$), all 8 functions are learned simultaneously in both systems without any incident. As depicted in Figure 2.9 B, above $5k\Omega$, the first breakdown values are obtained with the NOR function in both systems. Between $5k\Omega$ to $20k\Omega$, functions drop off sequentially, with 2 or 3 still persisting in correct output far beyond the ceiling value of $20k\Omega$. Neither hybrid nor all organic perform significantly better than the other- the lines closely resemble each other despite some deviations. This indicates that the all-organic automatic programming scheme demonstrated in Fig. 2.7)(B) does not require a major trade-off in learning performance.

Overall, both systems are very resilient to wire resistances. This is due to the fact that in all simulations initial conductance values are at G_{off}. As the final values exist within the range $1\mu S - 10\mu S$, this implies that even the most open memristors are still relatively resistive as compared to the wires. Yet, within the same order of magnitude (*e.g.* $R_{\text{mem}} = 100k\Omega$ while $R = 15k\Omega$), the voltage pulse needed to switch certain critical memristor devices may be attenuated enough that a threshold is no longer reached and the increments needed to implement supervised learning do not occur [204].

As seen in in Figure 2.9 C, some functions (eg NOR) break down early while others (eg NAND and IMP) do not. Moreover, functions that are inverse of each other (IMP, N-IMP; CONV, N-CONV) tend to do better preferentially between the two schemes. There is an underlying pattern: in the system using the organic memristive device in the neuron design, functions with

3 highs (1) and one low (0) are more resilient, while the opposite is true for the hybrid system using bipolar devices in the latch, (functions with 3 low and 1 high are more resilient). This is due to the differential maximal value of the latches. As the analog latch is more resistive overall ($G_{\text{off}} = 18\mu S$) than the binary latch ($G_{\text{off}} = 10\mu S$), the organic scheme should be better at clamping logic function with many low/'0' values even when wire resistances are very high. Inversely, as the G_{on} value of the binary device $G = 100nS$, is more conductive than the corresponding unipolar latch $G_{\text{on}} = 1.5\mu S$, it should better clamp logic functions with many '1' values when wire resistances are high.

Overall, the analysis suggests that parasitic wires are unlikely to force learning errors when the equivalent resistance of the least resistant state of the memristive devices G_{off} is still at least one order of magnitude more than the typical wire resistances.

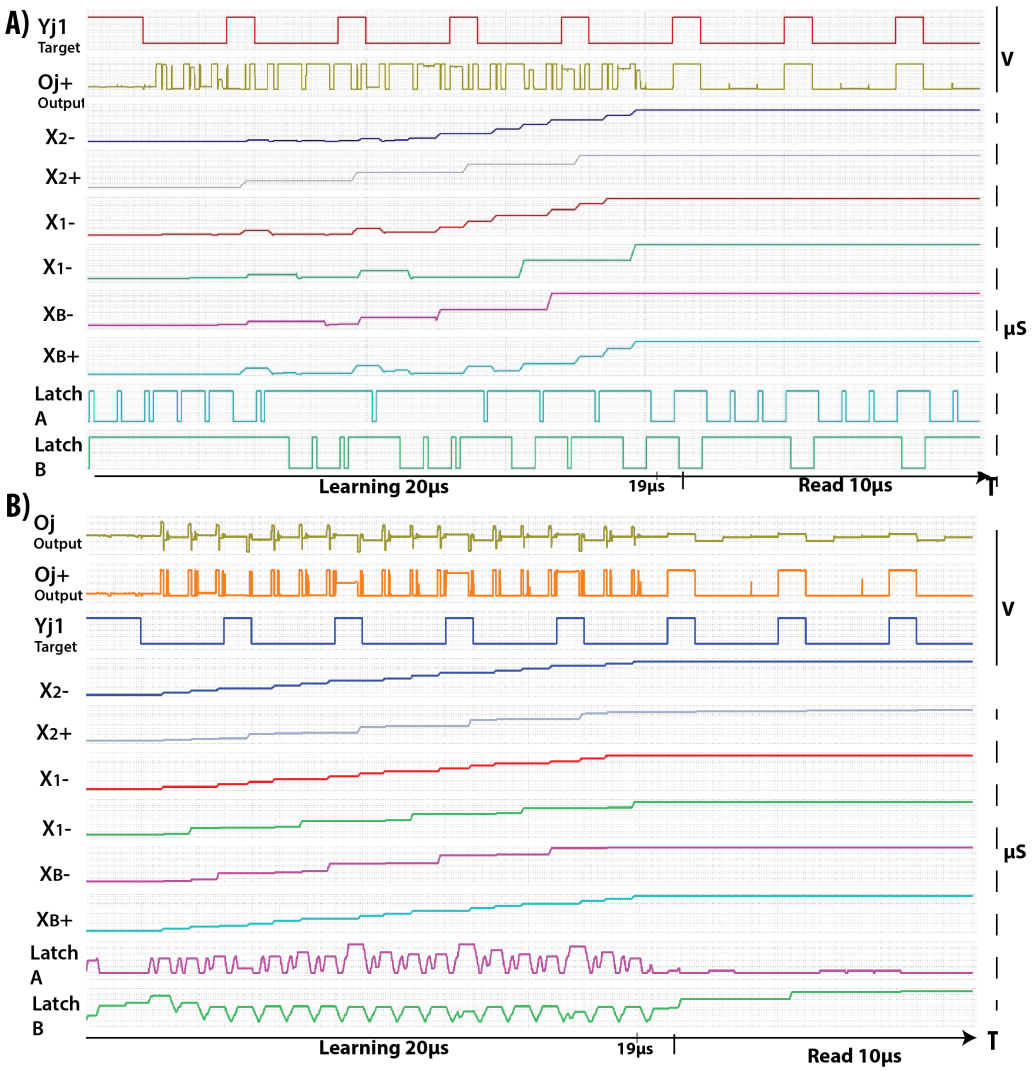
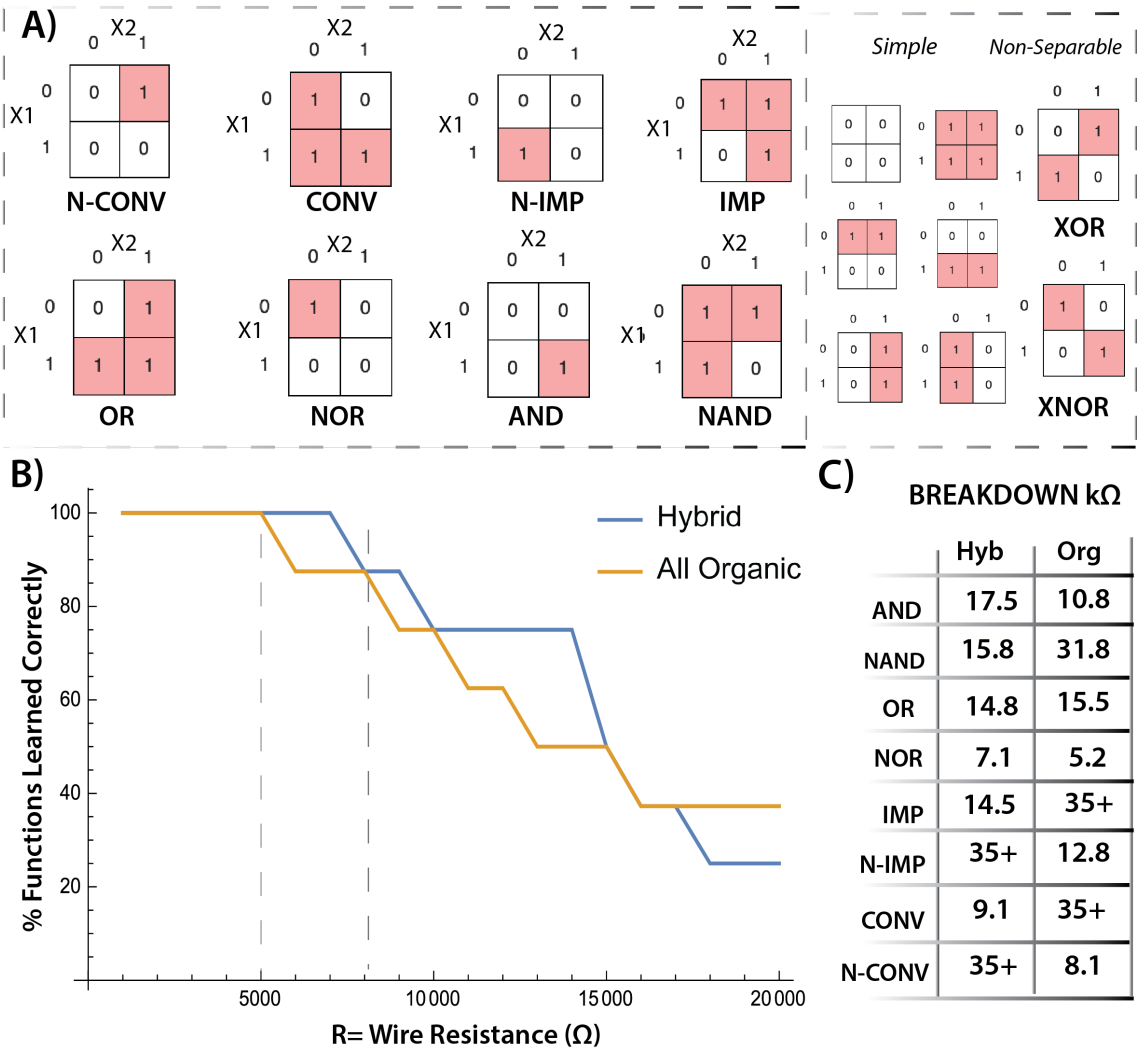


Figure 2.8: A) Waveforms depicting successful learning of the AND function in the Hybrid (organic unipolar memristors, binary latch) case; target, output, 6 analog memristors being trained, and latches are depicted from top to bottom. B) Equivalent waveforms for the All-unipolar organic memristor learning case. As visible Latch behavior is now also analog.



	Hyb	Org
AND	17.5	10.8
NAND	15.8	31.8
OR	14.8	15.5
NOR	7.1	5.2
IMP	14.5	35+
N-IMP	35+	12.8
CONV	9.1	35+
N-CONV	35+	8.1

Figure 2.9: A) All possible 2-bit functions. The 8 functions in the left box are the non-obvious ones on which the system is trained. B) Rate of success at learning these 8 functions for each variant system- hybrid and organic-, as a function of horizontal and vertical nanowire resistance R (40 parametric simulations), while R_{in} at the input wires is held constant at 500Ω. C) 'Breakdown' value in $R = k\Omega$: the given wire resistance at which the output for one of the four cases of the truth table is incorrect.

2.4 Development and utilization of a physics-oriented compact model

2.4.1 Motivation in developing/ evaluating a second model

The simple model used in Section 2.3 allowed us to gauge the potentials of our organic devices for learning in the context of a system level design (Neural unit framework). However, from the device perspective it had a few severe limitations:

- It did not consider the effect of initial (starting) conductance on the effect of the voltage pulse in changing the conductance (ΔG)
- Because it was not a time-parameterized model, it did not allow for an accurate understanding of the effects of pulse widths on conductance evolution
- The physical parameters used, such as α and β, were rough approximations of device behavior rather than being extracted from real devices via characterization procedures.

In this section, a physics-based compact model which is notably sensitive to the effects of both initial conductance and pulse timing (widths), is introduced and used to further test the resilience of our proposed learning systems.

From the point of view of contribution to the literature, we note that while a wide variety of inorganic memristor compact models have been published and many are available for researchers to build test circuits [207, 247–250], to the best of our knowledge no compact model of an organic memristive device is available to neuromorphic circuit designers. Thus an additional contribution of the following section is to contribute something back to the community.

The following compact model was developed in partnership with the Compact Modeling Team at IMS, Universite-Bordeaux, experts in the development of efficient analog compact device models, as part of the ANR-MOOREA project [237]. In addition, the electrical characterizations performed below were obtained via this collaboration.

2.4.2 Electrical characterization approach

The following results are derived from electrical characterization and parametric extractions performed on 33 tests structures which was performed at our partner lab, IMS Bordeaux. Ramp voltages ($0V \rightarrow 5V \rightarrow 0V \rightarrow 10V$) with a dynamic of 1V/s have been applied to the test structures. A typical I-V curve is shown in Fig. 2.10(a). At its initial steady state, the memristor is in a low conductance state. For low voltages between 0V to 3V (step 1), the current evolves linearly indicating that the memristor conductance remains stable. Once applied voltage rises above 3V, the current rises by at least two orders of magnitude as it approaches 5V (step 2). The applied voltage is afterward decreased from 5V back to 0V (step 3). The current measured remains

high indicating that the device state (ON) has been preserved. Finally, the voltage is increased yet again from 0V to 10V (step 4). Above the second threshold of 8.5V, current decreases by more than one order of magnitude and the memristor switches back to its initial (OFF) state. The same I-V characteristics were obtained with negative voltage ramps, confirming the unipolar (symmetric) nature of this particular organic memristor species. Three operating ranges are distinguished, with the two threshold voltages V_{ON} and V_{OFF}: READ mode, for which there is no variation of the conductance and the memristor's state follows the previous programming (voltage between 0V and 3V); SET mode, in which conductance evolves towards the conductive/ON state (voltage between 3V and 8.5V); and RESET mode, in which conductance evolves towards the insulating/OFF state (voltage above 8.5V).

We note that these are different, larger devices than the ones that produced the I/V curves in Fig. 2.1. In particular, the higher operating voltage range can be explained by the organic polymer dimensions considered for this study (junction area: 30nm x 250nm) [238]. As noted, this decreases when scaling to smaller organic polymer film thickness (Fig. 2.4); previously, it was V_{ON} = 3.5, V_{OFF} = 5.5. In addition to smaller dimensions, the scaling can bring additional trade-offs; as will be discussed in far more depth in Section 2.5, down-scaled devices notably suffer from larger variability than the devices studied here.

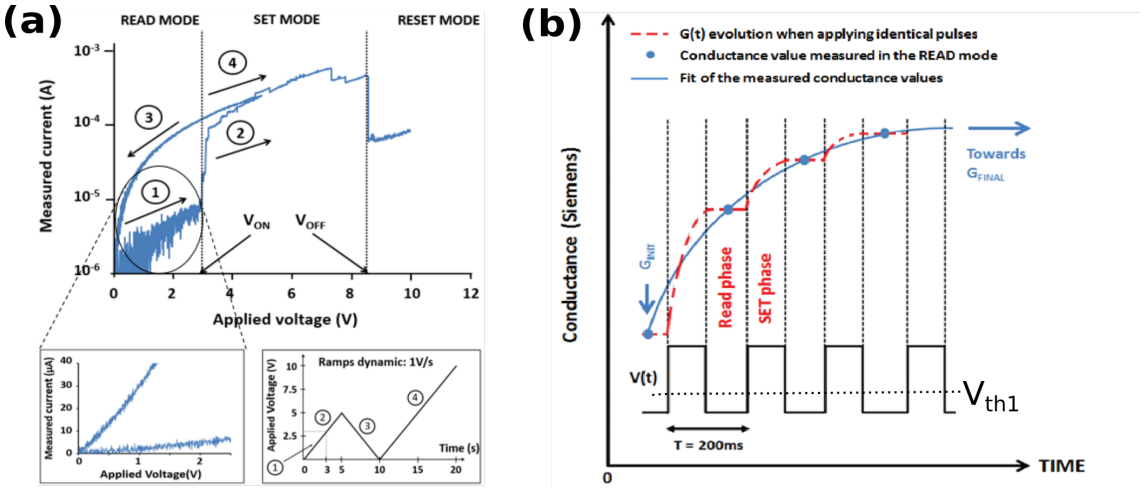


Figure 2.10: (a) Typical I-V characteristic in semi-log scale obtained for the test structures by applying, with a dynamics of 1V/s, the following ramp: $0V \rightarrow 5V \rightarrow 0 \rightarrow 10V$ (b) Schematic showing the evolution of the conductance when a train of n identical pulse are applied (red curve) and the conductance evolution obtained when fitting a first order response (blue curve) with the conductance values measured during READ mode (blue dots).

For these measurements, one hundred identical pulses of period T=200ms and 40ns rise and fall times were applied to the positive electrode while the negative electrode is grounded. During a pulse of width W, voltage is set to V_{HIGH} in which the memristor switches to the SET mode ($V_{ON} \leq V_{HIGH} < V_{OFF}$) or the RESET ($V_{HIGH} > V_{OFF}$) mode. The memristor conductance

value varies accordingly. During the moments between pulses (T-W), applied voltage V_{LOW} is within READ mode such that the memristor device retains earlier conductance value (Fig. 2.10(b)). The conductance evolution is obtained by measuring current when the memristor is in this mode. During the measurement process (20 seconds) the conductance evolves gradually each time a SET or RESET pulse is applied, from an initial value G_{Init} towards a final value G_{Final} after 20s. Once over, a series of pulses is applied to the memristor device's positive electrode to return it to conductance G_{Init}; this process is repeated ten times.

Several values of V_{HIGH}, W and G_{Init} have been considered during the measurement campaign. Parametrical extraction has been performed on various configurations given within the following ranges: $G_{\mathrm{Init}} = (5\mu S, 10\mu S, 50\mu S, 100\mu S)$; W = ($1\mu s$, $10\mu s$, $100\mu s$, 1ms); and V_{HIGH} varying from 2.4V to 10V with voltage step 0.2V. Depending primarily on the configuration (G_{Init}, W, V_{HIGH}), electrical behavior also varied between each of the ten individual runs at each configuration. This variation stems from delays and/or rapid conductance transitions that may be caused by: (i) intermixed filaments forming a more conductive single structure, (ii) several conductive paths located in different conductive areas and formed during a same pulse, (iii) or a conductive path formed by segments which have changed locally the conductivity.

2.4.3 Compact Model Approach

The conduction mechanisms have been modeled macroscopically by a filament-type mechanism for which memristor device conductance evolves depending on the number of conductive filaments created and/or destroyed. The conductance G(t) is assumed to be approximately proportional to the number of filaments and is assumed to be limited to a maximum value G_{Max} (G_{On}) corresponding to one unified/combined filament covering the entire device surface. Generation and destruction phenomena occur simultaneously, so conductance variation over time is:

$$\frac{dG}{dt} = S_1(G_{\max} - G(t)) - S_2 G(t)) \tag{2.4}$$

where S_1 and S_2 are respectively the mean conductive filament creation and destruction rates varying with time according to V(t), the time-varying voltage difference between the device's positive and negative electrodes:

$$S_1 = s_1 \exp(\frac{|V(t)|}{V_{o1}}) \tag{2.5}$$

$$S_2 = s_2 \exp(\frac{|V(t)|}{V_{o2}}) \tag{2.6}$$

The exponential terms represent the effects of the electric field on charge displacement between molecules having different energy levels, and V_{o1} V_{o2} are used as fitting parameters for this exponential expression. Since the model does not take into account the microscopic

geometry of the phenomenon, the electric field is summarized by the associated voltage V(t) between electrodes. In this equation, the generation rate takes into account the number of available positions for filament creation, with the $(G_{\text{Max}} - G(t))$ factor in Eq. 2.4. However, in the range of measured values, the extractions show that S_1 is very small in comparison with S_2 (equivalently, $G_{\text{Max}} \gg G(t)$) so that we cannot extract separately S_1 and G_{Max}, but only their product: z_1. Hence, the model is simplified when

$$z_1 = s_1 G_{max}$$

as:

$$\frac{dG}{dt} = z_1 \exp(\frac{|V(t)|}{V_{o1}}) - S_2 G(t) \exp(\frac{|V(t)|}{V_{o2}}) \tag{2.7}$$

If G_{max} varies significantly, which could be the case for devices at different scaling dimensions, parameter variation on z_1 explicitly can account for that. However, as our experimental characterizations revealed G_{max} variability in our properly programmed devices was not that severe (Fig. 2.11c), this parameter variation was not made a core part of our analytical approach.

2.4.3.1 Conductance Evolution Analysis

Conductance evolution analysis showed a first order system response trend in all configurations. For each configuration, ten runs mean conductance evolution G(t) have been considered and fitted; The parameter β is the inverse of the time constant (Eqn. 2.10), and parameters A and C are constant values defined in Eqns. 2.9, 2.11 respectively.

$$G(t) = A \exp(-\beta t) + C \tag{2.8}$$

$$A = G_{\text{init}} - G_{\text{final}} \tag{2.9}$$

$$\beta = \tau^{-1} \tag{2.10}$$

$$C = G_{\text{final}} \tag{2.11}$$

The device only evolves when it is in the SET or RESET pulse (width W), yet the entire timing window of analysis T is larger since it includes a period $(T - W)$ in READ mode $(T - W)$ where the device should be be non-volatile (Fig. 2.10(a)). Given this, Eqn. 2.7 is redefined as:

$$\frac{\Delta G}{T} = z_1 \exp(\frac{|V(t)|}{V_{o1}}) - s_2 G(t) \exp(\frac{|V(t)|}{V_{o2}}) \tag{2.12}$$

Table 2.1: Model Fitting Parameters. Units are the following:Vo_1,Vo_2, V; K_1, $S.s^-2$; K_2, s^-2; n_1,n_2 are unit-less.

Mode	Vo_1	Vo_2	K_1	K_2	n_1	n_2
Set	1.28	1.51	8.3.10$^-$6	8.1.10$^-$2	0.35	0.308
Reset	1.66	1.66	2.38.10$^-$6	0.5	0.35	0.308

This equation is a solution of Eqn. 2.8 and the coefficients of the two equations are associated as:

$$\alpha = z_1 \exp(\frac{|V(t)|}{V_{o1}}) \tag{2.13}$$

$$\beta = s_2 \exp(\frac{|V(t)|}{V_{o2}}) \tag{2.14}$$

$$G_{\text{final}} = C = \frac{\alpha}{\beta} = \frac{z_1}{s_2} exp(\frac{|V(t)|}{V_{o1}} - \frac{|V(t)|}{V_{o2}}) \tag{2.15}$$

The parameters z_1, s_2, V_{o1}, and V_{o2} were extracted by parametric analysis on the mean conductance evolutions G(t) according to Eqns. 2.9,2.12 of all the (G_{Init}, W, V_{HIGH}) configurations.The extractions revealed that parameters V_{o1} and V_{o2} might be considered as constant. In contrast, parameters z_1 and s_2 depend on the pulse width W. Indeed, the number of filaments created and destroyed increases for longer W. Parameters z_1 and s_2 are expressed as:

$$s_1 = K_1 W^{n1} \tag{2.16}$$

$$s_2 = K_2 W^{n2} \tag{2.17}$$

Values V_{o1}, V_{o2}, K_1, K_2, $n1$ and $n2$ are listed in Table 1 for both SET and the RESET modes.

2.4.3.2 Integration in Compact Model

The $Fe(bpy)_3^{2+}$ organic memristor compact model has been implemented in Verilog-A. It is a three port structure having two I/O ports (“in” and “out”) corresponding to the organic device’s positive and negative electrodes, and a third port Gf where the current value of the conductance is stored and read. Current is calculated as:

$$I_{\text{in,out}}(t) = G(t)(V_{\text{in}}(t) - V_{\text{out}}(t)) = G(t)V_{\text{in,out}}(t) \tag{2.18}$$

The only user defined input to the compact model is an initial or starting conductance G_{Init} at t = 0. Then, conductance variation is calculated according to Eqn. 2.8, which in turn depends on the pulse width W given by Eqns. 2.16,2.17. The model itself evaluates the pulse width during a transient simulation when $V_{\text{ON}} \leq V_{\text{in,out}}(t) < V_{\text{OFF}}$ (SET mode) or $V_{\text{OFF}} < V_{\text{in,out}}(t)$ (RESET mode). Equation 2.12 is therefore rewritten as:

$$\Delta G = X_1 W^{n1} - X_2 G(t) W^{n2} \tag{2.19}$$

$$X_1 = K_1 T \exp(\frac{|V(t)|}{V_{o1}}) \tag{2.20}$$

$$X_2 = K_2 T \exp(\frac{|V(t)|}{V_{o2}}) \tag{2.21}$$

Given that the integral on conductance variation over the range 0 to W is:

$$\int_W^0 \frac{dG}{dt} dt = \Delta G - 0 = \Delta G \tag{2.22}$$

Conductance variation as function of time is finally:

$$\frac{dG}{dt} = n_1 X_1 t^{n1-1} - n_2 X_2 G(t) t^{2n-1} \tag{2.23}$$

In the Verilog-A description, conductance evolution for a given time window G(t) corresponds to a floating voltage V_{Gf} taken between a current generator delivering a current equal to (18) and a capacitor C integrating the same current. Hence, the capacitor C retains the device's conductance value. The state variable G also takes the value of V_{Gf}. Current intensity is then calculated as a function of the voltage dynamics $V_{\text{in,out}}(t)$, variable G and simulation time t. At the initial step of the simulation ($t = 0$), the compact model is at $V_{\text{Gf}} = G = G_{\text{Init}}$

2.4.3.3 Comparison between Model and experiments: SET

For configurations at higher W, a higher dispersion exists between the G_{final} values than in the lower ones; for instance, the largest dispersion ($\sigma = 16.9\mu S$) comes when $W = 1ms$ (Fig. 2.11b). This is because, in these cases, the results of the longer programming event are more unpredictable, as many filaments could be created or destroyed simultaneously during it duration. This can result in very different effects depending on G_{Init}, and even lead to incomplete filament formation issues (hence, some devices never reach G_{max}). In contrast, when W is far smaller, programming is a more well-controlled phenomenon, resulting in a more predictable synaptic outcomes. As visible in Fig. 2.11c, where $W = 100\mu s$, almost all devices in the more gradual SET mode reach their maximal values, resulting in a fall smaller ($\sigma = 4.8\mu S$) dispersion value. Since increased dispersion counteracts desired model determinism, this favors shorter and more predictable pulse programming modes for our later neuromorphic applications. As visible in Fig. 2.12a, our analytical expression is finally in agreement with normalized device behavior (averaged across all configurations).

2.4.3.4 Comparison between Model and experiments: RESET

The analytical expression is also validated in the RESET mode relative to all measured conductance evolutions. Regardless of W, dispersion is more prominent in the RESET mode than in the SET mode. All of the conductance transitions for one configuration within RESET mode are visible in Fig. 2.11a. In most of the investigated configurations, conductance decreases abruptly after the first pulse having an amplitude higher than V_{OFF}. It implies, when correlating measured G(t) with Eqn.2.6 , that the time constant (Eqn. 2.10) between the initial conductance and the final conductance is less than W. This relates to the fact that conductive paths in the RESET mode can be entirely or partially destroyed. One hypothesis is that although the pulse dynamics are rapid, since $V_{\mathrm{OFF}} > V_{\mathrm{ON}}$, conducting paths are recreated during the rise and fall times because of inrush currents. After this definitive step, conductance variations stabilize and oscillate around a mean value. As already shown in the I-V graph of Fig. 2.10(a), conductance stabilizes in the RESET mode after its initial decrease at 8.5V. Fig. 2.12b reiterates this behavior.

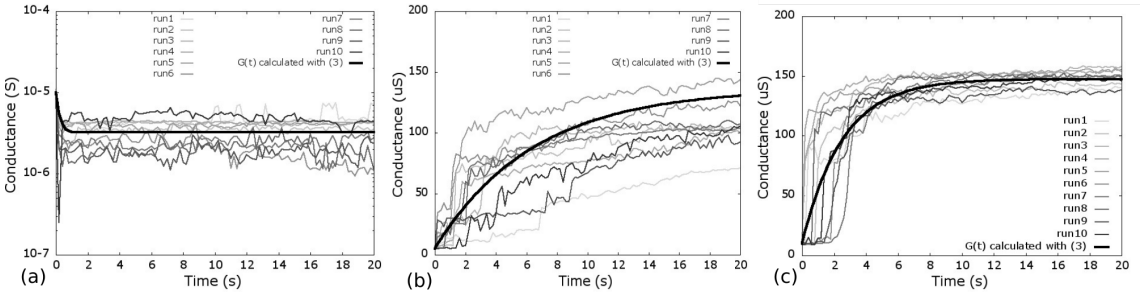


Figure 2.11: (a) Analytical and actual conductance evolution compared for RESET for one configuration: $G_{\mathrm{init}} = 10\mu S$, $W = 100\mu s$, $V_{\mathrm{HIGH}} = 8.6V$ (b) Similar comparison for SET mode, $G_{\mathrm{init}} = 10\mu S$, $W = 1mS$, $V_{\mathrm{HIGH}} = 8.6V$ (c) A second set configuration, but with $W = 100\mu s$

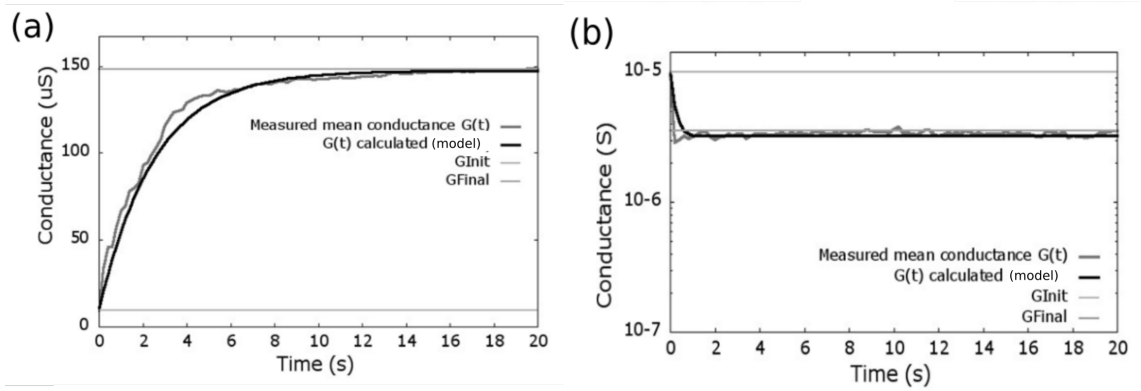


Figure 2.12: Normalized conductance evolution measurements during *all* successive pulse runs, for both SET (a) and RESET (b) modes, with the compact model's prediction according to (18).

2.4.4 Systems-level scripting methods

For use in neuromorphic applications with complex learning procedures, a versatile compact model ought to keep its conductance state between individual learning stages. For this reason, we have extended the compact model to a dynamical approach through the use of a complementary software tool chain assessing circuit description organization, visible below conceptually in Fig. 2.13. This approach is especially useful for circuit electrical schematics that contain a large number of memristor compact model instances, e.g., large crossbar architectures.

The dynamical approach uses a PERL script to process the post-simulation data retrieved after simulation of a circuit netlist in Spectre format (.scs file). Post-simulation data is retrieved in the .raw format- a table of all the voltages and the currents of a considered circuit integrating instances of the compact model. Among all of these voltages and currents, the PERL script retrieves, for each compact model instance, the image voltage corresponding to G(t) by identifying the voltage coming from the compact model Gf port, *i.e.* V_{Gf}. The process has been automated by integrating the model as a sub-circuit. The PERL script generates a file to be included in the netlist which updates G_{Init} values for each memristor compact model instance contained in the circuit description using .ALTER instructions. In continuing to simulate with Spectre software, the circuit netlist now includes all previous conductance values; the G_{Final} values from the previous simulation have become the G_{Init} values of the next one. Still, repeating this method manually may be tiresome. For this reason, a shell script which performs each of the individual steps was written to automate the process. The functions of this script are visible as the gray surrounding box in the right pane of Fig. 2.13. Now, the user needs simply output a circuit netlist (design) to the home directory, specify the desired number of iterations, and invoke the bash script. The script edits the netlists via the awk Unix command and outputs n text files into a results folder which correspond to the final conductance at the end of each epoch.

2.4.5 Considered architecture

Our system is composed of several individual neural learning units, as described in Section 1.4.5. In the following sections, we consider these systems in an unsupervised (plasticity oriented) context, as well as in the context of a discrete supervised learning task.

2.4.5.1 Neuron design

In contrast to the hybrid neuron design selected, for the following demonstrations we built an entirely CMOS neuron system with the crucial component being a differential amplifier built using Cadence's analog library. As is standard, the non-inverting input is grounded and the inverting input is connected to the entire line, as visible in the bottom right of Fig. 2.14. This ensures that sum of all input voltages multiplied by the respective conductances $M_1, \ldots, M_n$, as given by Kirchoff's Law, produces a unitary current output, and allows our circuit to perform

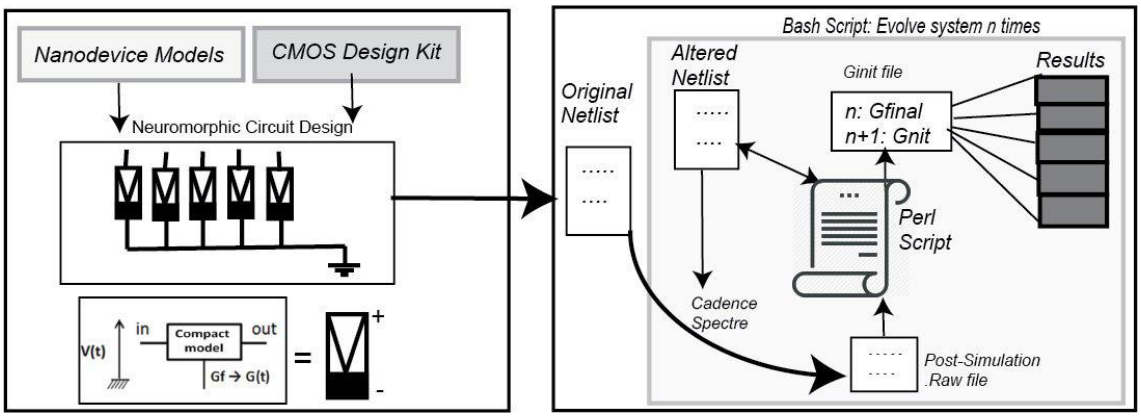


Figure 2.13: Left pane: shows integration of compact model and standard CMOS design kits into a static design. Right pane: shows the dynamic method developed where the design (netlist) can become an evolving neuromemristive circuit using the proposed scripting techniques.

approximate dot-product operations on-chip [220]. The pre-neuron current value is multiplied by R, effectively converting small current changes in Pre into much larger voltage changes at Post. An additional benefit of the amplifier design is that the output varies in the power range and is constant outside of it, making it appropriate for both analog and digital signals being passed forward in a mulitlayer systems [251]. A transistor acts as a switch to apply learning pulses during correct moments only, so that the circuit is stable and weights can be read in between active (programming) moments; this allows for stable and fast online learning.

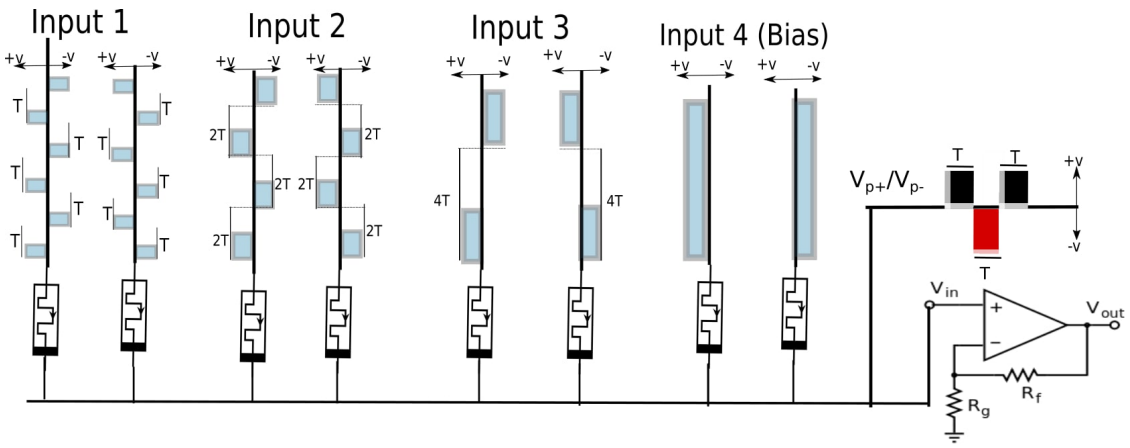


Figure 2.14: This electrical schematic demonstrates the system used to produce all results in Sec. 2.4.6. The system is built entirely out of CMOS components and instances of the organic memristor devices (represented here as the memristor circuit symbol).

2.4.6 Unsupervised adaptation in a single neural learner

Before attempting a supervised learning task, we first analyzed the adaptation abilities of a single neural learner consisting of 4 input channels (8 input wires total) connecting to a single common (output) line via 8 model organic devices. To create an environment for differential synaptic adaptation, we set each of the pre-synaptic CMOS neurons to send pulses at different frequencies or rates; this approach relates to a critical observation in neuroscience, which is that rate-based encoding of spikes may be critical to information encoding, decoding, or both in neural circuits [98, 252]. Hereafter, T is a global timing parameter used to scale the overall rate of learning by setting the period of input (pre-synaptic) and feedback (post-synaptic) voltage pulses. The pulse programming scheme chosen to demonstrate emergent plasticity effects, completely visible in Fig. 2.14, is the following: inputs oscillate between $V_{\text{in+}} = 1.5V$, $V_{\text{in-}} = -1.5V$, at multiples of period T, with half of the period an active pulse (W). The parameter additionally varies line-by-line; for devices fed by Input 1, pulse width is $\frac{T}{2}$ and period T; devices fed by Input 2, period $2T$ and pulse width T; devices fed by Input 3 have period $4T$ and pulse width $2T$; positive and negative devices receive input patterns at opposite moments as each other. Bias inputs are constant with the positive line always at $V_{\text{in+}}$ and the negative always at $V_{\text{in-}}$. Programming pulses applied on the common line (Post) oscillate between $V_{\text{p+}} = -4.6V$, $V_{\text{p-}} = 4.6V$, with both period and width T; this programs devices in either SET or RESET mode, depending on the combination of voltages. Note that, in order to make this workable, V_{OFF} was edited from $8.1V$ to $6V$ in the compact model description (it does not otherwise change the model functionality).

2.4.6.1 Sensitivity to initial conductances

Fig. 2.15(a)-(c) shows the evolution of three systems with the previously described differential encoding set-up, where the only difference between the three panes is starting conductances. Electrically, since the voltage for each device is shifting as a combination of the sign of the input as well as the voltage drop between input and shared wire (Pre), dynamic collective behavior exists between the devices and their evolution through time is quite complex, as visible in the many small oscillations. Each device demonstrates plasticity by adapting its preferred state towards the combination of the respective input line it is connected to and the post line. The speed of the adaptation is non-linear; that is, more changes occur towards the first half of the simulation (between 0 and 5ms) than in the latter half (5ms to 10ms). Since, as visible, the devices are not all saturated at the minimal (G_{min}) or maximal (G_{max}) values, further investigations were made as to how the rate of adaptation or plasticity additionally relates to the global timing parameter (T).

2.4.6.2 Sensitivity to pulse width

Now, the same circuit and task was considered but varying timing parameter (T) used to set the overall pace of adaptation across the systems. As the total simulation time is always capped to $t = 20ms$, this creates a trade-off between frequency and intensity or programming/plasticity events; Fig. 2.15(d)-(f) demonstrates three possible outcomes. As visible, when T is quite large (f), fewer total programming or plasticity events occur, and steps are larger and far more dramatic. This results in slower convergence overall, and somewhat unpredictable learning outcomes. Conversely T is very small (d), adaptation happens very rapidly and smoothly due to the many small increments; this creates a relatively predictable dynamic, but there is less richness in the plasticity effects. The middle timing case (e) shows an intermediate case where the weights adapt in a complex way, with most converging towards a final state, but some oscillating. Note, finally, that pulse programming regime has an important effect on the outcome of the final synaptic values; for instance, Input 3+ ends up in maximal, minimal, and intermediate conductance levels depending on whether the global timing parameter was fast (d), intermediate (e), or slow (f), respectively.

These experiments demonstrate the ability of the neural learner electrical system to act as a laboratory for examining plasticity effects in nanodevices. For instance, the observed state-dependent and rate-dependent plasticity behaviors could be exploited in unsupervised or semi-supervised learning schemes. Weight-dependent STDP was recently demonstrated using pulsed biasing on metallic oxide nanodevices [253]; promisingly, our organic nanodevice model also evolve in a state-dependent way due to differential voltage biases.

2.4.7 Demonstration of sequential simulations

In this section, we show the implementation of the scripting methods of Sec. 2.4.4 using the simple single learner circuit (*i.e.*, the netlist used to obtain these results is the same as what produced those in Figs. 2.15 (a)-(f)). Given this constant input system, $i = 10$ iterations are performed for one of three different cases, where what is varied is again the global timing parameter T. The results for ten subsequent cascading iterations are visible in Fig. 2.16 (a)-(c), where (a) shows the case of $T = 10\mu s$, (b) $T = 100\mu s$, and (c) $T = 500\mu s$; note that the first epoch in each chart corresponds to final resting states in Fig. 2.15(d)-(f). This demonstration shows how complete adaption or learning is necessary in order to evaluate the real potential of a scheme. While in this set-up the fast system (a) converges completely within the first two epochs, the slow system (c) converges between epoch six to ten, depending on device. In the context of this device, it suggests that an intermediate timing parameter (between $T = 100\mu s$ and $T = 500\mu s$) may be optimal such that convergence does not happen either too fast or too slow. In the context of our proposed simulation methodology, the numerical efficiency of the model and developed scripting tools will be an asset to neuromorphic hardware designers looking for quick and electrically realistic evaluations of learning architectures. In the context of application to

neural network set-ups, these simulations highlights that weights may drift in an unintended context if convergence speed is not set properly; this would be even more crucial to analyze in a larger network which may require dozens or even hundreds of epochs. In the next section, we evaluate one relatively simple yet typical neural network set-up.

2.4.8 Electrical Simulations: Small digits

Finally, we examine how the compact model and neuromorphic circuit building blocks can be integrated to perform a toy data science task. We implement an on-chip learning system that classifies digits from the *scikit-learn* small digit database [254], also called the OCR database. This database consists of 1797 examples of each of the 10 digits; each has a dimensionality of 64 grey-scale pixels (individual training images are 8x8) (Fig. 2.17 (b)). To achieve this we built a second, far larger crossbar schematic in SPICE which connects 130 pre-synaptic neurons to 10 outputs via 1300 organic memristive compact model instances (while the digit database have a dimensionality of $n = 64$, double lines and two bias lines are required as per the neural learner scheme). Unlike in the logic function implementation previously detailed in Section 2.3.4.1, the presentation of images through these channels is not multiplexed in time, but instead training images are presented simultaneously in their entirety. The digit database was split into training and testing cohorts of 1200 and 597, respectively. Neural network training was conducted during several epochs; during each, one quarter (300) of the total training database was randomly selected and randomly ordered before presentation to the network.

During one training step within a given epoch, a known digit is presented in voltage mode in the sign-symmetric fashion shown in Fig. 2.17 (a); immediately after, corresponding error cases are corrected in each weight pair by applying the appropriate polarity of V_p to the post-synaptic line of each of the systems in error; this follows the implementation given in Section 1.4.5 and corresponds to a single iteration of stochastic gradient descent. The width of this pulse now corresponds to the timing or learning rate earlier discussed: $W = T$. As before, $V_{\text{p+}} = -4.6V$, $V_{\text{p-}} = 4.6V$, $V_{\text{in+}} = 1.5$, $V_{\text{in-}} = 1.5$, to manifest the conditional programming scheme during training. New for this case, input voltages are analog and correspond to pixel values in inference/test mode. Weight updates and learning were all done in-situ in the Cadence platform. After all epochs have concluded, all unknown digits are presented sequentially using a script and guesses, defined as the maximal output or most confident prediction, are compared to true values in order to yield a final classification success percentage. An example of one inference step during the testing period is visible in Fig. 2.17(e), where the read-out vector is produced as a the dot-product of input digit's voltage input vector and the crossbar's weights. In contrast to the un-supervised case analyzed earlier, the configuration of initial weights is not critical to the functioning of the algorithm, although this does introduce variability in learning outcomes.

2.4.9 On-chip learning outcomes

2.4.9.1 Loss from ideal software outcomes

When convergence is successful, our OCR learning system built with the organic compact model instances achieves greater than 90% of the accuracy that could be achieved with a comparable software ANN. When successful, the nanodevice weight matrix manifests one hyperplane in $\mathbb{R}^n$ space (where n is again the dimensionality of the task) for each of the separately learning neural units; in this case 10 units, one trained to recognize each digit against the others, learn in parallel. We achieve mean 89.8% classification in this optimally converging range. In order to demonstrate why such a strong classification can be made with just linear classification boundaries, we performed a clustering analysis of 500 examples from the training set with t-distributed stochastic neighbor embedding (t-SNE), a machine learning technique which is ideal for mapping datasets from high-dimensions back to a visually understandable 2-dimensional space [255]. As visible in Fig. 2.17 (c), linear classification in the high-dimensional space should indeed be able to separate the vast majority of class examples from each other, with some notable exceptions, e.g. '1' class which blends with several others. To further test this hypothesis we trained a support vector machine in software, which also trains to use a hyperplane for classification. This software system achieves between 93 – 97% classification, as listed in Table 2.2. The 4 – 7% loss in accuracy from the software ideal is mostly accounted for due to the effect of our device's characteristic non-linearities; while software learning systems use perfectly linear weight updates, SET and RESET produce different ΔG values in our device, which may also be scaled according to the device's starting conductance G_{init}. This effect has previously been discussed in the context of PCM devices [256].

2.4.9.2 Convergence depends on adaptation speed

However, successful on-chip learning with the present algorithm is not always possible, and indeed heavily dependent on the pulse width parameter $W = T$. While $200\mu s < T < 400\mu s$ showed strong results, pulse programming widths below this range resulted in poor performance due to the synaptic saturation effects already discussed, and pulse programming above it resulted in slow or failed convergence. The two faulty cases are highlighted in Fig. 2.16 (e), and Fig. 2.16 (d), respectively, which show the evolution of 32 devices within one neural learner during training. Examples for all three cases are demonstrated in Table 2.2, with standard deviation noted where each case was attempted 10 times with different initial random conductance values in the on-chip array. As visible, the effect of the first failure case is far worse than the second case, in which learning is mostly correct just incomplete.

2.4.10 Implementing Elementary Learning Rates

The dynamic approach allows one to both inherit final conductances from a previous simulation and to change key learning parameters in the simulation in between sub-steps (epochs) of a complex learning procedure. As a further demonstration of the value of the dynamic scripting

Table 2.2: Performance on the Image Classification Task (%)

	$100\mu s$	$250\mu s$	$500ms$	Software (SVM)
OCR: Mean	62.5	89.8	78.2	95.2
OCR: Deviation	3.85	2.35	6.21	1.52

approach, an elementary form of learning rate modulation was attempted in the standard on-chip procedure by changing the pulse width W in the midst of a multi-epoch learning process. As visible in Fig. 2.16 (f), while the approach succeeded in adding some form of momentum, e.g., forcing it to converge towards the end of learning, the end results were not very different than the 'slow' system learning case overall. Moreover, the achieved results with this approach (mean 87%) were no better than the a constant optimal W value throughout training. This suggests a more complex scheme is necessary; while a wide variety of more complex options are available to optimize stochastic gradient descent in software systems [257], their on-chip implementations may be complex and remains a topic for future consideration.

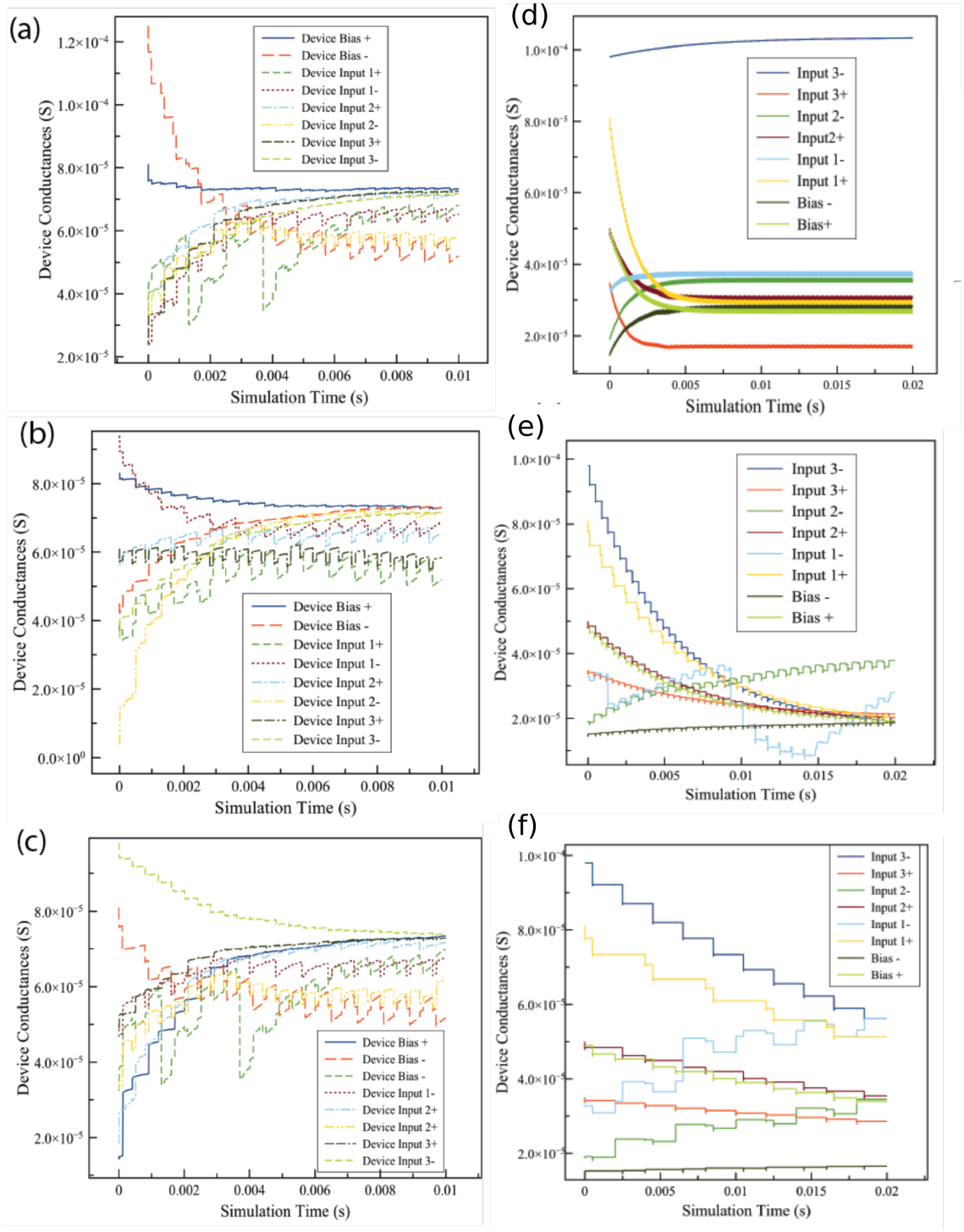


Figure 2.15: (a), (b), (c): Three identical simulations depict the evolution of 8 compact model devices during a simulation period $t = 10ms$ according to the differential pulse programming scheme described in the text. The only difference between the three panes is the initial set of device conductances. (d),(e),(f): systems are initialized at equivalent conductances, while pulse width is being varied instead. (d) shows the case of $T = 10\mu s$, (e) $T = 100\mu s$, and (f) $T = 500\mu s$. Total simulation time in these cases is now $t = 20ms$.

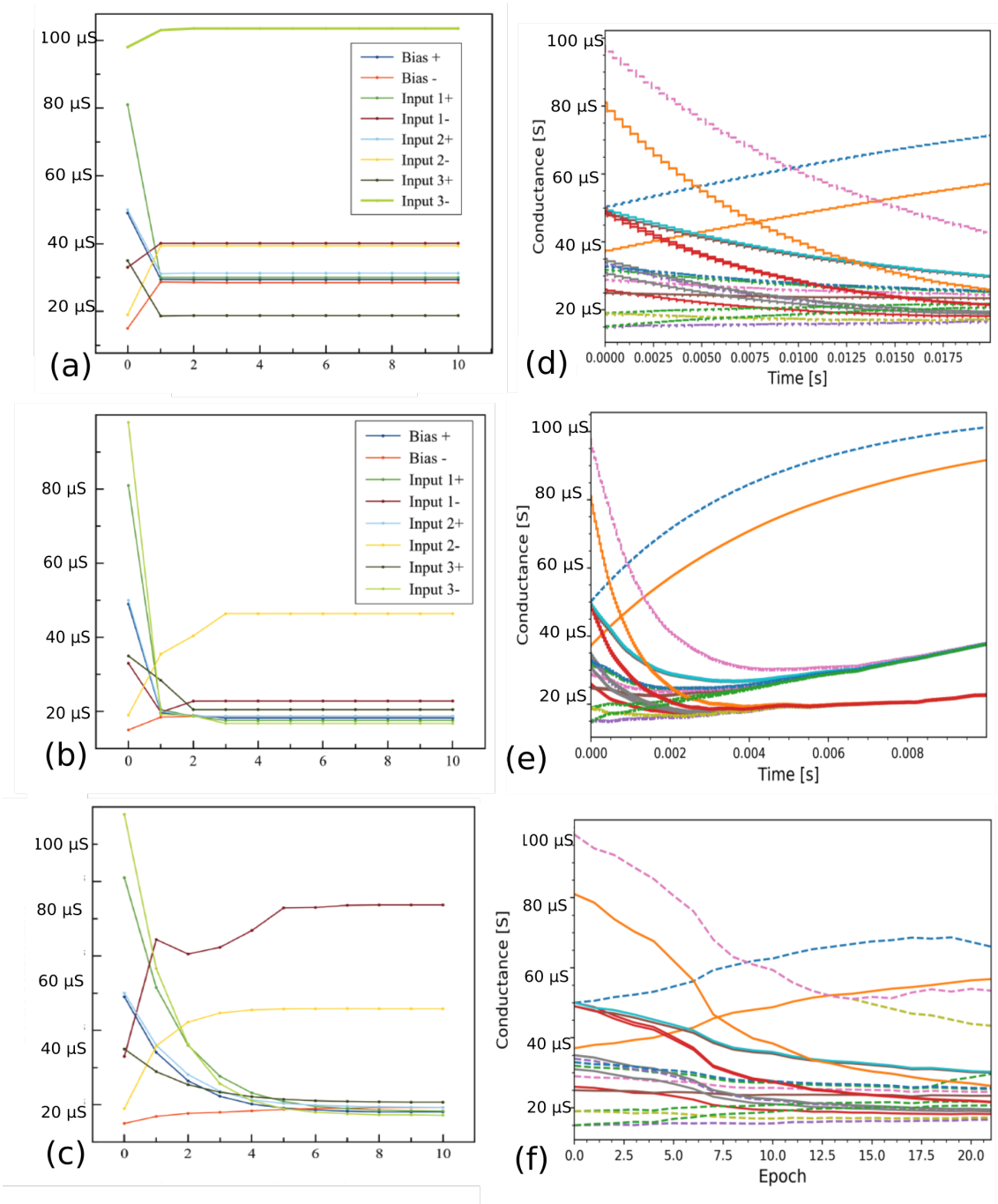


Figure 2.16: (a)-(c): demonstrate sequential simulations. At step 0, all conductances are initialized randomly. After every step, all conductances values evolve based on previous G_{Init} values. Note that the transition between weights from $epoch = 0$ and $epoch = 1$ is precisely the transitions shown in Fig. 2.15(d)-(f). Total simulation time of each epoch is $T = 20ms$.
(d)-(f): track evolution of 32 memristor compact models connected to the same output neuron during training. (d)'Large' pulse ($T = 500\mu s$) learning takes a long time to converge ; (e) 'small' pulse ($T = 10\mu s$) learning, seems to learn *too* quickly and converge weights too close to each other; (f) The dynamic scripting tools are used to change T in a multi-epoch learning environment; $T = 500\mu s$ during the first 6 epochs, $T = 100\mu s$ during the second 6, and $T = 10\mu s$ during the last 8.

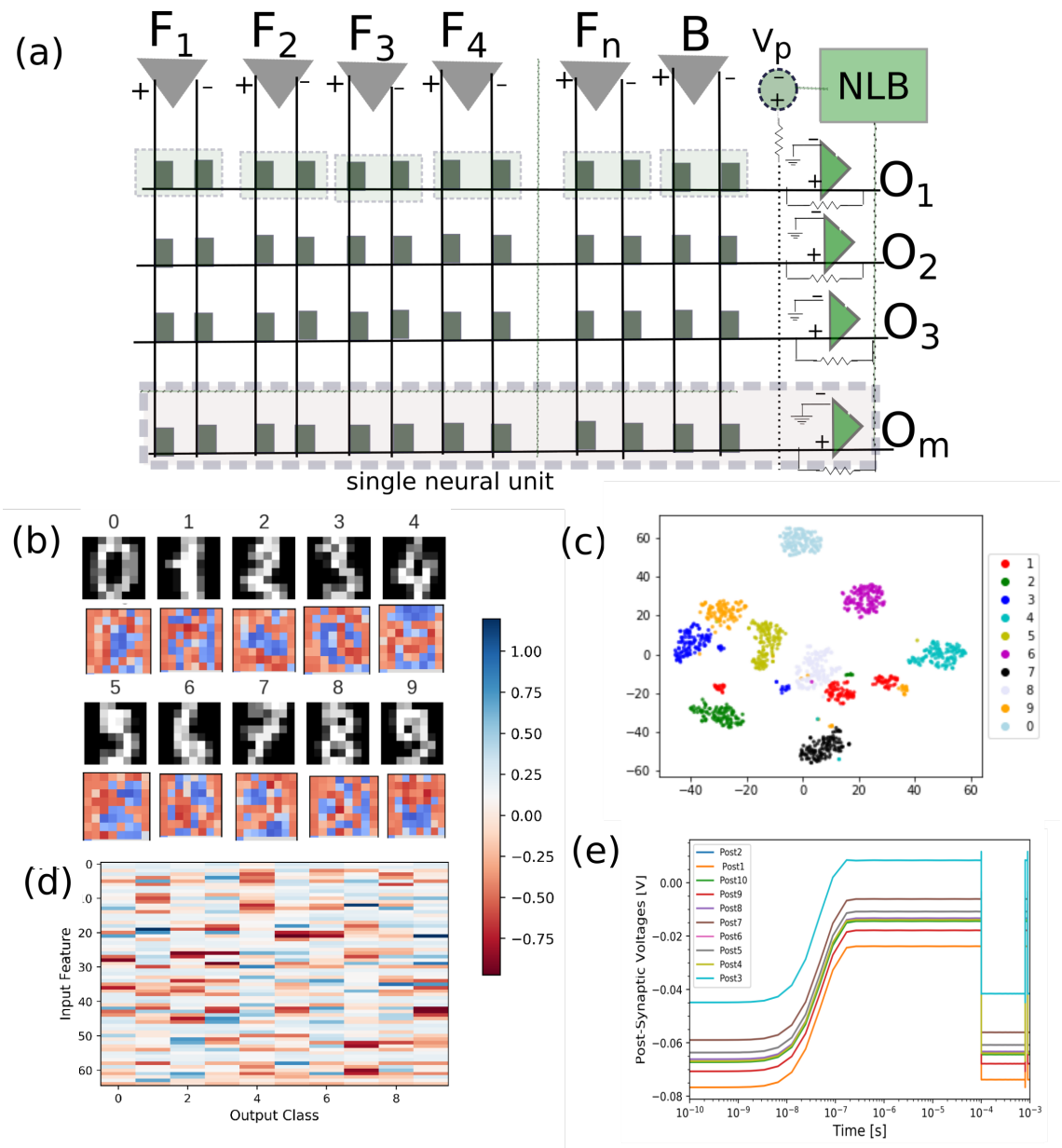


Figure 2.17: (a) The simulated crossbar neuromorphic architecture is illustrated; it expresses differential weights connecting inputs to outputs, which are in turn connected to a learning cell and/or future layers (b) A single example from each of the ten classes from the Sci-Kit digit database is shown alongside reconstructed neuron-by-neuron 'weight filters', that correspond to each of the ten output neurons of our model system; (c) Sci-Kit Digits/OCR dataset is visualized in arbitrary two dimensional space using the Barnes-Hutt implementation of the t-SNE mapping algorithm; each point is an example and its color is the class, as in the legend; axes are arbitrary or unit-less (d) The entire weights matrix obtained using the on-chip learning algorithm are demonstrated; colorbar, shared by both, shows all weights as normalized pairs (-1 corresponds to $G_{min} - G_{max}$; 1 corresponds to $G_{max} - G_{min}$; (e) During the entire time duration, a single training digit (in this case, corresponding to the class '3') biases the network at $0.1V$, producing the pictured post-synaptic voltage on lines O_1 to O_10. During an active 'fire' moment' (between $t = 10^-7$ and $t = 10^-4$), a companion pulse carries only the 'winning' neuron above the threshold (0), implementing a 'spike' carried forward to the next layer.

2.5 Experimental Demonstration of Scheme

2.5.1 Motivation

There are several inter-related challenges in moving from device simulations to real hardware prototypes: intrinsic variability of memristive devices may be worse than expected, non-ideal circuit behavior may disturb theoretically perfect learning schemes, and devices may behave dramatically different in 'live use' than their characterizations or models would suggests. The primary motivation in building and realizing learning in the following system, was to enhance our understanding both at the device and systems levels of the feasibility of on-chip (local) learning with emerging filamentary nanodevices. By discovering more about these factors, we also contribute to the community of neuromorphic engineers, because there are just a few works that address in-depth imperfect behaviors in physical learning systems [59, 256]. While some of the phenomenon addressed in these works are irrelevant to our case (resistance instability, which is only applicable to phase change devices), others, such as an asymmetric increase (SET) and decrease (RESET) of device conductance in filamentary-based devices and stuck-on/off effects, are very relevant to our case. We extend this sort of analysis to a completely novel, polymeric filamentary device, which should be very helpful for future designers of new any new variety of synaptic organic nanodevice. Overall, in the following work we discuss and demonstrate ways to improve tolerances of generic learning rules for real-world systems.

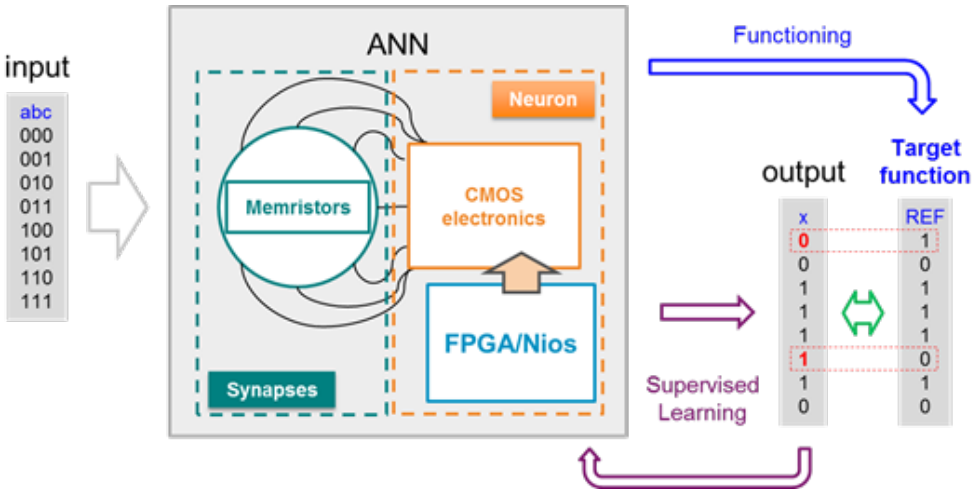


Figure 2.18: This high-level depiction of our set-up demonstrates the key components of our ANN system, including memristive synapses, CMOS accompanying circuitry, and off-chip programming set-up. The overall task is to have the ANN reconfigure itself to reproduce (output) any chosen binary teacher (input) signal. Because we only use one learner system, however, a major limitation of the present set-up is that this teacher system must be linearly separable.

2.5.2 Building an Adaline System with our devices

In this work, we have experimentally realized a single learner system or Adaline with 4 synapses (8 organic memristive nanodevices). We have realized experimentally a full ANN set-up that uses accompanying CMOS logic devices to instruct the nanosynapses to perform online learning autonomously. The circuit is composed of a CMOS-based neuron implemented on a field-programmable gate array (FPGA), and a series of memristive devices mimicking synapses between differential inputs and the neuron, as demonstrated in Fig. 2.18. The circuit encodes weights using signed-synaptic pairs of memristive devices with modifiable conductivity; $n+1$ pairs, or $2n+2$ memristive devices, are required to successfully map a function with n inputs. Each of the $n+1$ inputs requires a negative and positive wire to separate states for that case, and negative and positive bias lines configure the entire line. In response to a set of input voltages, the sum of all conductances is proportional to current on the common post-synaptic line. Precisely, given inputs $X_1 \cdots X_{n+1}$, a shared post-synaptic potential X_j is obtained automatically by linear combination of pair weights:

$$X_j = \sum_{i=1}^{n+1} = W_i X i \tag{2.24}$$

X_j is converted from current to voltage and from analog to digital ($sign(V_j)$) after it passes through a comparator set to ground. This has the effect of inverting the sign of X_j. For every (2^n, where n is the number of bits) case of the function's truth table, the learning block checks every post-comparator's output (O_j) to verify if it is the same sign as expected (Y_j). If so, the next case is checked; else, if $sign(Y_j) \neq sign(O_j)$, a programming pulse is sent. In this case, the learning block applies an appropriate programming pulse, as defined by pre-defined logic in the FPGA. Once trained for all cases of the target function, our system should be able to reproduce a desired signal based on its synaptic weights and input signals.

As previously described in Section 1.4.5 generically, and in Section 2.3.4.1 specifically for our device, our system learns using an online, binary adaptation of the Widrow-Hoff rule (WH). This requires it to learn example-by-example using a canonical procedure. Specifically, at each step (epoch), difference between expected and actual output s computed and is compared on the basis of the $sign$ function and an appropriate adjustment is made to minimize that cost.

In this experimental work, we expanded upon the earlier scheme based on the insight that our multi-threshold device can, when needed, offer a choice between two programming modes (Fig. 2.19c). As visible in Fig. 2.19c-d, first threshold programming uses only SET mode of the device (hereafter, Set Only (SO) mode), or two of the four active programming steps. Since the polarity of a programming pulse follows the line output O_j, $V_{p+} = V_{t1+}, V_{p-} = V_{t1-}$ in SO mode. Two threshold programming uses both SET and RESET (hereafter, Set Reset (SR) mode), implements all four error-correcting steps with just two pulses as shown in Fig. 2.19e (and, as previously visible in Fig. 2.5). Again because conductance falls across the second threshold, and rises above the first, $V_{p+} = V_{t2-}, V_{p-} = V_{t2+}$ in SR mode. In practice, then, SO and SR

modes send pulses with opposite voltage polarity to correct an equivalent error. The precise voltage programming configuration to implement both SR/SO modes in our experimental work is listed below in Section 2.5.3.2.

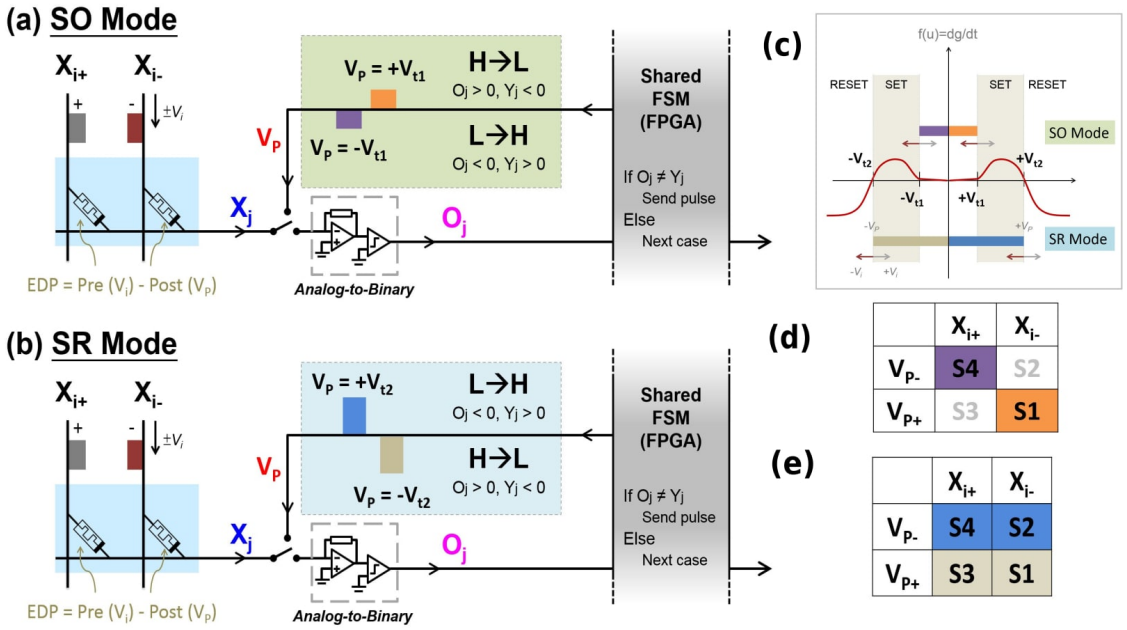


Figure 2.19: (a) A schematic of SO programming in an active case; while both possible correcting pulses are shown on the line, only one would be sent corresponding to depicted error case (b) Similar schematic showing SR programming being used to correct an error. (c) A diagram of device conductance evolution as it relates to appropriate thresholds for programming pulses in both modes. (d) Color-coded table of the active steps that SO programming implements. (e) Color-coded table of SR programming that implements all active steps. (f) Table which shows input, expected, line output, and prescribed weight change binary (sign) values at each of the four active steps.

2.5.3 Experimental Set-up

2.5.3.1 Overview of Operation

The electrografted memristive synapses have been integrated in a chip with 22 total devices (11 on two individual lines), where they can be accessed individually and collectively on each of their ports using individual input lines along with the common line. Once put in vacuum using an accessory pump (Alcatel ACP 286), said chip is connected to a custom-designed printed circuit board (PCB) connected to a power source (Agilent E3631A). Within the PCB, devices are connected to accessory circuitry such as the comparator, current to voltage converter, etc needed to read line output. In addition, the PCB contains components for electrostatic discharge (ESD) protection. The board is directly connected to an FGPA (Altera Cyclone DE2-70),

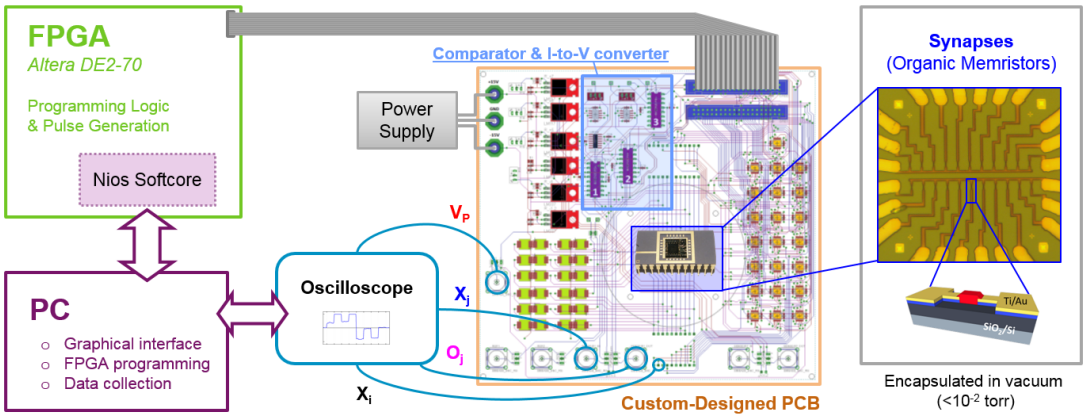


Figure 2.20: Organic memristive nanodevices, placed within a custom designed PCB board connected to a power source, are put into vacuum using an accessory pump. Within the PCB board, the devices are connected to accessory circuitry (comparator, current to voltage converter), needed to interpret line output. The PCB board contains additional components for electrostatic discharge (ESD) protection. The board is connected to an FGPA, which sends appropriate programming pulses using the logic (FSM) it was programmed with before the learning process began using a PC. A NIOS softcore was created to allow for real time user control of the functions that have been loaded onto the FPGA during programming via the connected PC interface. Functions such as erase, read, and learning modes (single epoch, or continuous) can be applied to devices subsequently. Finally, an oscilloscope probes the key electrical ports noted in Figure 3,4: line output X_j, comparator output O_j, programming pulses V_p, and expected function Y_j. The oscilloscope is also connected to the PC for real time data collection.

which both reads from the devices and sends appropriate programming pulses by using the onboard logic (FSM) with which it has been programmed by accompanying generic Altera software (Quartus). A custom NIOS softcore dedicated to interface with a PC and corresponding graphical user interface were coded to allow for real time user control of the functions that have been loaded onto the FPGA during programming. Functions such as erase, read, and learning modes (single epoch, or continuous) can be applied to devices subsequently. An oscilloscope (Agilent MSO 6014A) probes key electrical ports and is also connected to the PC for real time data collection assisted by LabView. A representation of the actual setup is below (Fig. 2.20).

2.5.3.2 Voltages scheme implementing learning

Table 2.3 shows the actual voltage levels used to manifest conductance change according to the conditional logic scheme just described. As visible, the electrical potential across every device satisfactorily implements the active error-correcting steps.

	Error case	Actually applied voltages (V)					Individual conductance (G) and Pair Weights Changes			Step(s)
		X_{i+}	X_{i-}	V_P	EDP+	EDP-	ΔG_+	ΔG_-	ΔW_{ij}	
SO Mode	1 (H→L)	+1.5	-1.5	+3*	-1.5	-4.5	-	↗	-1 (↘)	S1
	1 (L→H)	+1.5	-1.5	-3	4.5	1.5	↗	-	1 (↗)	S4
SR Mode	1 (H→L)	+1.5	-1.5	-5*	6.5	4.5	↘	↗	-1 (↘)	S3,S1
	1 (L→H)	+1.5	-1.5	+5	-4.5	-6.5	↗	↘	1 (↗)	S4,S2

*Correspond to V_{P+} in Fig. 3d,e of the main text. (The unmarked ones correspond to V_{P-})

Table 2.3: Simplified example of each programming case as presented in Fig. 3 of the main text, assuming V_t1=3V, V_t2=5V, V_i=1.5V, and electrical differential potential across the memristor (EDP) = $V_i - V_P$. X_{i+} and X_{i-} stand for the input of one memristor pair, S1-S4 are active step to correct errors according to the WH table, as also shown in Fig. 3 of the main text.

2.5.4 Learning Results

Both SO and SR programming schemes demonstrate successful learning of diverse 3-input functions using 4 pairs of organic memristive devices (3 pairs for each of the input lines, and one pair for bias).

2.5.4.1 Characteristic Learning Experience

One learning example using SO scheme is presented in Fig. 2.21a-d. In this case, the system is learning the "A nand B and C" function. To read the initial state of the neural learner, a series of input signals is sequentially applied at 10kHz rate, representing the 8 different 3-input configurations (i.e. 000, 001, ..., 111). The devices can be programmed with pulses as short as 1 μs, but with such short pulses, programming cannot be considered reliable. For this reason, in this demonstration, 100 μs pulses are chosen. The blue line in Fig. 2.21a is the output of current-voltage converter (noted schematically on Fig. 2.19), which represents the total post-synaptic weight (X_j) of all memristor pairs. Note that as depicted in Fig. 4 the post-synaptic value (blue line) depicted is always inverted (-X_j) due to the operation of the transimpedance amplifier. The pink line shows the output of the comparator (O_j), which compares actual X_j (blue) to ground (0V). If $X_j > 0$, O_j is pulled towards "high" output (1); if $X_j < 0$ it is pulled to "low" output (0). As shown in Fig. 2.21a, the initial state of the neural learner gives an output of "00110011" from the eight (A, B, C) input configurations.

Figure 2.21b shows the synaptic weight evolution (top panel), and error counts (bottom panel) at each epoch. Errors are gradually corrected until the system reaches an error-free state after 7 epochs. Fig. 2.21c shows an example of every event inside one learning epoch. The black line is the input of one memristor, X_i+, changing its sign according to the input signal at positive polarity. The blue and pink waveforms- X_j and O_j respectively- are read from the same nodes as depicted in Fig. 2.19a,b. The red line indicates a measurement along the wire that supplies programming pulses (V_p as noted on Fig. 2.19a,b). It shows that three pulses were

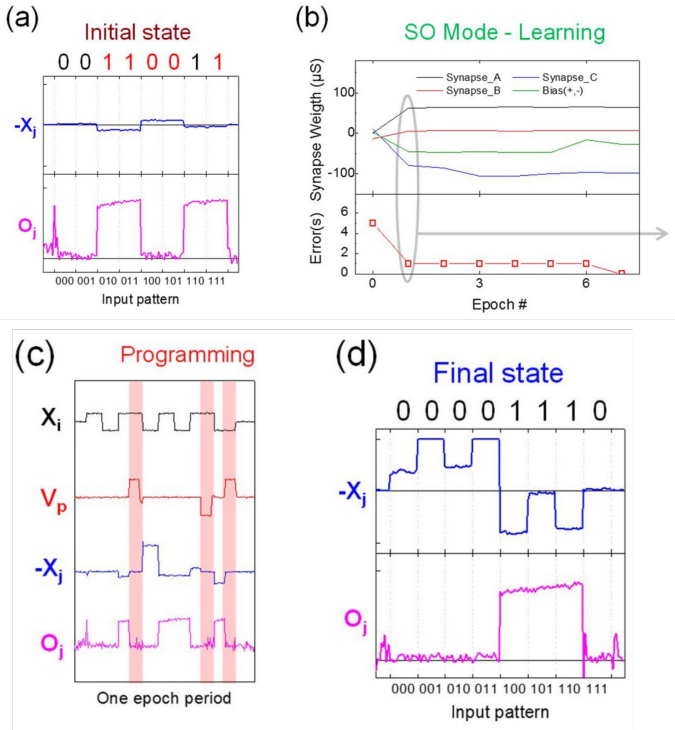


Figure 2.21: Learning of "A nand B and C" function (00001110) using the SO programming mode. (a) Output of I-to-V converter (blue line, -X_j) and comparator (pink line, O_j) showing the initial state of the system of each learning. The initial errors are marked in red. (b) Learning histogram showing the synaptic weights (top panel) and total errors (bottom panel) evolution at each epoch. (c) Example of a single learning epoch (marked with grey circle in (b)) showing the input X_i (black), programming pulses at Y_j (red), synaptic output -X_j of the I-to-V converter (blue), and digital output of the comparator O_j (pink) which is being compared to Y_j. The active programming steps, when the system attempt to correct an error, are shaded red. (d) System output at the end of the learning, showing successful learning of the "A nand B and C" function.

applied in this particular epoch to correct the output errors for "010", "110" and "111" inputs, respectively. It should be noted that when not in programming mode, a switch guarantees the common line is virtually grounded by the current converter, as also shown in Fig. 2.19a,b. Fig. 2.21d probes the synaptic output (X_j) and digital output (O_j) nodes at this final state. It clearly shows that the CMOS neuron has learned the target function, "A nand B and C", by producing the output "00001110" when provided a truth table as input.

Fig. 2.22a-d shows a learning example using the SR mode instead. Error counts starts at 5, and oscillates thereafter until the function is learned perfectly at epoch 13. The synaptic weights are adjusted actively during the learning process to reach the final state of "A nand B and C" function. As visible, the main difference between SO and SR programming styles is that there is much less fluctuation of synaptic weight during learning in the former. This is because SO programming uses only SET, which changes a given memristive device's conductivity more

gently than RESET. It should be noted that SR programming can begin regardless of the initial state. Whilst, for SO learning, all memristive devices were first RESET before learning begins (since a physical decrease in device conductance is not accessible through this learning rule).

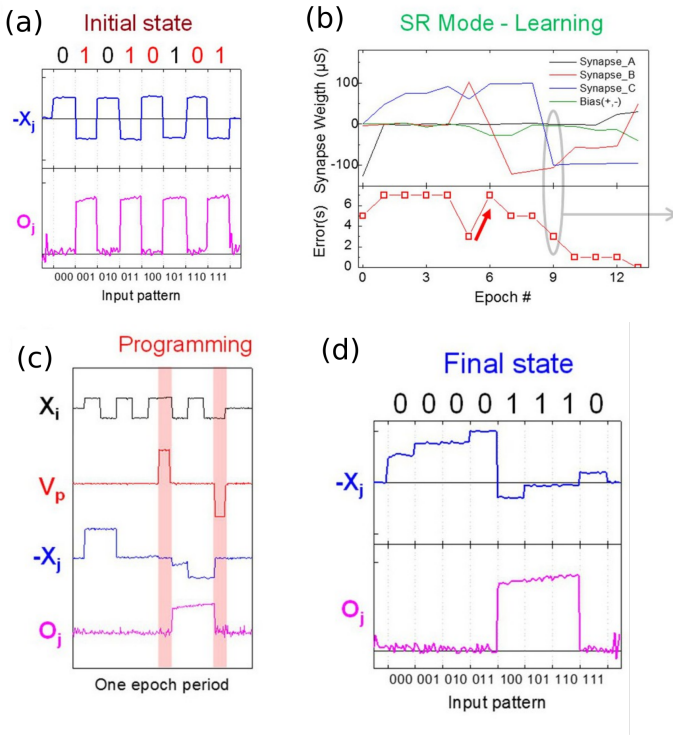


Figure 2.22: Learning of "A nand B and C" function (00001110) using the SR programming mode. Panels are the same as in Fig. 2.21.

2.5.4.2 Overall learning performances

In total, 7 linearly-separable 3-bit functions were attempted by our demonstrator using both programming styles (SO,SR). Of the 7 functions attempted, 5 were successfully learned in both cases (a 71% success rate). All successful cases are noted below for SO and SR modes in Tables 2.4 and 2.5, respectively. Although a very small sample, mean values show that SR completes faster but 'wastes' many pulses in the process. Conversely, SO takes more epochs but less pulses. The former is explained by RESET overshoots; the latter, by 'sticky' devices that take many epochs to reach a high enough conductance.

2.5.4.3 Resilience to Imperfect Devices

Large device variability is a major setback toward the realization of robust ANNs. Fig. 2.23a and b show the typical V_{t1}, V_{t2}, G_{Max}, and G_{Min} variability of 11 devices (in the same row). The variation of V_{t1} is relatively small compared to that of V_{t2}. This suggests the learning system should be more reliable if only V_{t1} is required for learning (SO mode). As for G_{Max} and G_{Min},

	Epoch Learned	Errors Corrected
3NAND	33	38
(A and B) or C	21	31
A → (B → C)	9	26
A nand B or C	7	10
MAJ	12	24
Mean	**16.2**	**25.8**

Table 2.4: All Functions learned Successfully by demonstrator using First Threshold (SO) Programming

	Epoch Learned	Errors Corrected
MIN	14	39
A nand B or C	13	64
MAJ	6	13
A and (B or C)	8	20
3AND	22	39
Mean	**12.6**	**35**

Table 2.5: All Functions learned Successfully by demonstrator using Second Threshold (SR) Programming

their variations are relatively large, which is a common issue in memristive devices. Nevertheless, a relatively wide working region exists, as shown in the green zones of Fig. 2.23b, which permits learning in the system. The immediate effect of variability is to increase the number of epochs required to learn. For example, when using G_{Min} as initial states, devices with lower G_{Min} need extra time to correct their errors, while a lower G_{Max} will reduce the safe working range. The measured variability of all 11 devices on chip were 10%, 14% and 59% for V_{t1}, V_{t2}, G_{Max}, respectively, as summarized in Table 2.6. Our learning demonstration was carried out using the 8 most similar devices, reducing G_{Max} variability to 40%.

	Mean	Standard deviation	Variability
V_{t1} (V)	3.78	0.39	10%
V_{t2} (V)	6.72	0.97	14%
G_{Max} (μS)	69.5	41.2	59%
*G_{Max} (μS)	73.3	29.0	40%

*G_{Max} of the 8 memristors used in learning system prototype.

Table 2.6: Typical variability of memristive performances extracted from 11 devices of the same chip.

Although less studied, physical devices possess non-idealities beyond variable response to equivalent voltage inputs. In the case of our organic-composing device, two additional effects-asymmetric switching behavior and evolution of threshold voltages through time- have non-negligent effects on our supervised-learning system. Asymmetric behavior manifests as an imbalance between the SET/RESET processes of our device. To increase conductivity, filaments

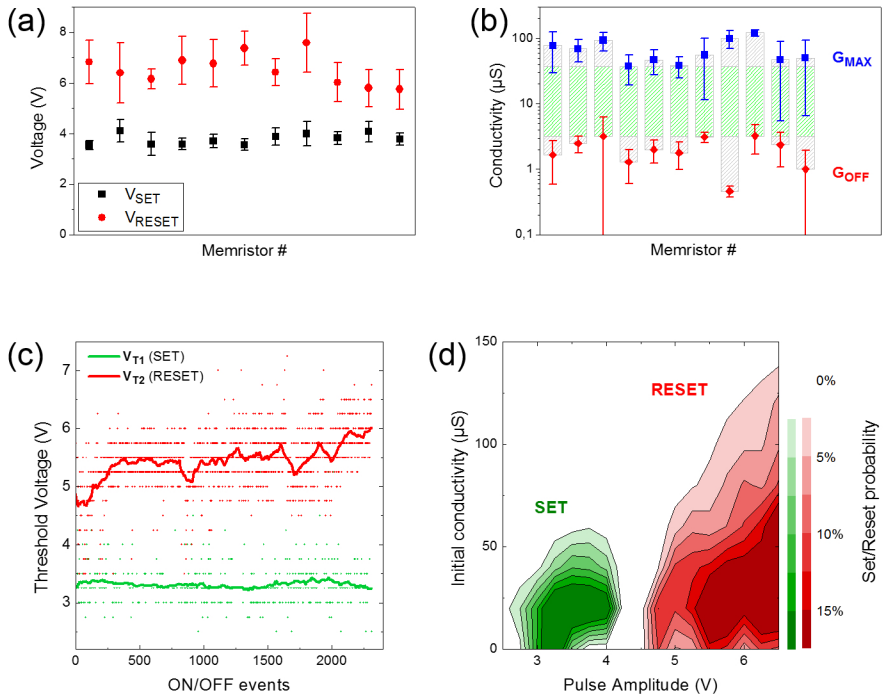


Figure 2.23: (a) Threshold voltages variability of 11 memristors in the same chip (V_{t1} in black and V_{t2} in red). (b) Maximum and minimum conductivity (G_{Max}, G_{Min}) variation of these 11 devices. The symbols marks the average values and the error bars indicate their standard deviations. (c) Evolution of SET and RESET events in numbers of On/Off cycles, extracted from the data shown in Fig. 2.2b. The green line shows the moving average of V_{t1}, and the red line for the V_{t2}. (d) Contour map of SET and RESET voltage with respect to their initial conductivities, G_0.

build up gradually through atomic/ion diffusion or charge transfer/trapping. By contrast, the decrease of conductivity is mostly caused by the breaking of the conductive filament, which is a violent process. This asymmetric behavior causes a fundamental problem during SR learning: instead of gradually approaching the target output (decreasing errors) at the constant weight adjustment required by WH, a dramatic RESET overshoots and in turn creates more errors than it corrects (red arrow in Fig. 2.22b). An immediate way to avoid this issue is to switch learning algorithm: in SO programming, memristive devices are programmed only at the first threshold, avoiding dramatic RESETs. Inversely, devices with the opposite SET/RESET asymmetry could use Reset-Only programming to avoid dramatic SET [81]. To prevent conductances from saturating (vanishing) in such a scheme, a RESET (SET) pulse would be required at every pair after a certain number of unsuccessful learning cycles.

The second physical constraint encountered with our devices is their evolution through operating time. Figure 2.23c shows the evolution of SET and RESET events in sequentially applied On/Off cycles. While V_{t1} remains constant after 2300 cycles, V_{t2}, experiences non-negligible changes throughout the measurement period. It tends to gradually shift to higher voltage,

drops to a lower voltage, and begins another upward drift cyclically. The observed behavior is compatible with formation and growth of conductive filaments. As the filament grows thicker, larger current (thus higher voltage) is required to break it. As also shown in Fig. 2.23d, the RESET threshold increases for a higher initial conductance. At some point, when the broken filament can no longer recover, a new thin filament is grown, which breaks at lower voltage. The aging of the device makes formation of subsequent filaments harder, and provides an explanation for why V_{t2} generally moves to a higher voltage over time. This impacts operation of the learning system, as the device can no longer RESET if it does not dynamically adjust the amplitude of programming pulses. While lifetime can eventually be improved by engineering, one immediate way to reduce aging is again adjustment of the learning algorithm. Switching from SR to SO programming reduces the number of On/Off events per active cycle by half (since the former implements every active step at each cycle while the latter keeps half devices in read mode at each cycle). When considering the combined effect of violent RESET and double switching activity, a switch from SR to SO could increase device and thus system lifetime substantially.

Device imperfections also affect the efficiency of SO learning, similarly increasing number of required epochs for learning. As shown in Fig. 2.21b, the number of errors may remain unchanged for several epochs before decreasing again. This is due to non-linearity in the SET mode: while a programming pulse efficiently increase conductivity (ΔG)of the devices at lower conductance, the increment is reduced when the state prior to a pulse is already conductive. While this does not prevent learning, it does decrease efficiency. If both devices in a synaptic pair reach maximum conductivity in SO mode, i.e. are stuck at the ON state, the only solution is to RESET the system and start another learning cycle, which can complicate the function of the neuron. Moreover, if a single device is thoroughly stuck-on, learning may not be possible depending on the pair it is in and function being learned.

2.5.5 Complementary Simulations evaluating effect of variability

2.5.5.1 Simulation Methodology

We expanded upon the generic model presented earlier in Section 2.3.1 to develop a model which individualizes the evolution of each device in an array where $G_{\max}$, $G_{\min}$, and the corresponding conductance change ΔG is thus different for each organic memristor device. Assuming the device is biased at a voltage V_b above the first critical threshold V_{th1} but below V_{th2} and decreases above V_{th2}, and given a total number of total addressable states g and the write range , $r = G_{\max} - G_{\min}$, as follows :

$$\frac{dG}{dt} = \begin{cases} V_{th2} > |V_b| > V_{th1} & f(|V_{b1} - V_{th1}|, g, r) \\ |V_b| > V_{th2} & -f(|V_{b2} - V_{th2}|, g, r) \\ |V_b| < V_{th1} & 0. \end{cases} \tag{2.25}$$

In our case where dt is set, e.g we employed a constant programming scheme with $V_{b2} = V_p+ = 5.5V$ for reset, $V_{b1} = V_p- = 3.3V$ and programming (pulse) length $dt = 100\mu s$, $\Delta G = f(g,r)$ can be approximated in a discrete time-step simulator during each writing (learning) phase as follows:

$$\Delta G = \frac{r}{g-1}. \tag{2.26}$$

We always set $g = 64$, thus variability is introduced through the r parameter as constructed by the device's individualized writable range. For the control case of perfect devices, every device has identical extrema ($G_{\max} = 69.5\mu S, G_{\min} = 2.1\mu S$) and identical thresholds; for the variable cases, we have introduced extrema distributed around a Gaussian spread from those means, and with first and second thresholds distributed around $V_{\text{th1}} = 3.1$, $V_{\text{th2}} = 5.5$. Finally, to account for the asymmetry issues, we have treated the ΔG_{SET} as a baseline for each device, and multiplied $\Delta G_{\text{RESET}} = \gamma \Delta G_{\text{SET}}$ where γ is the asymmetry parameter. This model was built and evaluated in Matlab.

2.5.5.2 Impact of Device Imperfections on Logic tasks

First, we used the model to attempt the same logic-learning functions we attempted experimentally. For each function being learned, 500 Monte-Carlo iterations- each which begins with a random set of low initial conductances among 8 simulated memristive devices- are simulated. When variability mode is enacted, at each iteration, inter-device variability is emulated by picking random first threshold, G_{Max} , and G_{Min} values for each memristive device from a normal distribution around a characteristic value with $\sigma(G_{\text{Max}}) = 40\%$, $\sigma(V_{\text{t1}}) = 10\%$, and for SR simulations, $\sigma(V_{\text{t2}}) = 10\%$. Even in the default case, $\gamma = 1.33$, reflecting the fact that RESET is more powerful than SET. Every simulated system in a given iteration is granted 50 epochs; if it does not solve all cases of the target function's truth table by the final epoch using WH to adjust weights, learning is considered a failure. Average success rate is given by the number of successful trials divided by iterations. Errors corrected and epoch completed are obtained by taking mean values over all iterations (if the function failed to learn, epoch learned in that case is counted as the maximum (50)).

The results of these functional simulations stress the decisiveness of device variability and conductance change asymmetry. Without variability in device thresholds and G_{Max}, learning is always possible; the 7 three bit functions learned by our demonstrator learn in simulation every time, by mean 3.5 epochs for SO programming and mean 4.4 epochs for SR programming (when characteristic change in conductance above the first or second thresholds, $\Delta G_+, \Delta G_-$, respectively, are constant for every device). However, variability creates imperfect learning outcomes. Table 2.7 lists averages for 500 trials with variable nanodevices in SO mode. As visible also in Fig. B.3, the average success rate now varies slightly function-by-function. SR programming introduces the possibility of an asymmetry between the sizes of the characteristic conductance change $\Delta G_+, \Delta G_-$. Table 2.8 (also Fig.B.4 (a)) shows the variable simulations

for this result in a mild asymmetry case (ΔG_- is slightly larger than ΔG_+).

Simulated learning results for SO, SR modes are similar: every function is learned successfully in at least four out of five cases at experimental levels of variability, while SO is slightly faster. Demonstrator learning results (Supplementary Tables S2,S3) show that SR mode actually finishes faster on average (mean 12.6 epochs, SR; mean 16 epochs, SO), while requiring more programming pulses (35 SR, 26 SO). Although the sample size of successful demonstrator examples is small, characteristic device imperfections are nonetheless highlighted by this contrast. In SR mode, dramatic RESETs double the error pulses predicted by the model to be sent in one case. While SR mean epoch nearly matches predicted, SO mean epoch is driven up by functions that struggle against non-linear conductivity: 2 functions in particular (3NAND, "(A and B)or C") each take 10 epochs to correct their final error case, nearly doubling epochs from 9 to 16.

Our simulations also highlight the fragility of SR mode by exploring the effect of the asymmetry parameter, γ. While mild intra-device asymmetry as in Table 2.8 is workable, stronger asymmetry ($\Delta G_- \gg \Delta G_+$) produces an average success rate of only $5\% - 30\%$, depending on the function (Fig. B.4 (b)). When device conductances remain 'pinned' to low values, insufficient weights are available to separate some cases of the truth tables of some functions from others (Fig. B.4 (e)). The converse strong asymmetry case ($\Delta G_+ \gg \Delta G_-$) yields better results for the SR style than any other configuration (Fig. B.4(c),(f)) because it has the opposite effect of increasing the span of possible weights (conductances).

Complex, non-linearly separable functions may be perfectly learned in several layers when our system is cascaded, or imperfectly learned in one layer with larger neural units than the one we physically realized. To show the former, we emulated a multi-layer perceptron system by feeding forward functions learned in a first layer to build the truth tables of linearly non-separable functions in subsequent layers [117]. Similarly to [258], which showed a multi-layer memristive system can learn AND and NOT in a first layer with two neural learners and subsequently the 2-bit XOR function with another in the second, we learned the 3-bit XOR function (01101001). Three learners in the first layer and one in the second (32 organic memristive devices total) are required to resolve this problem. The function is perfectly learned with perfect devices; when approximating experimental levels of inter-device variability, both programming styles can still successfully resolve the problem: mean 79% success rate is obtained for SR mode (Fig. B.5 (a),(c)) and 71% for SO mode (Fig. B.5 (b),(d)).

2.5.6 Demonstration on larger image task

Lastly, we used this model to simulate the performance of a hypothetical crossbar of our devices performing online learning in the context of a canonical image recognition task (the MNIST database of handwritten digits [159]). To do this, we simulated crossbar composed of 15,680 of our organic memristive synapses. Ten separate perceptrons or neural learners- one for each digit class- each require double the number of synaptic devices as pixels ($p = 784$) to

	Avg Success	Avg Epoch	Avg Errors
3NAND	87%	9.36	25.33
(A and B) or C	80%	12.77	36.73
A → (B → C)	88%	9.05	30.02
A nand B or C	86%	9.98	28.1
MAJ	91%	8.54	25.08
A and (B or C)	83%	11.35	32.76
3AND	85%	10.73	30.1

Table 2.7: Simulated results for First Threshold (SO) Learning assuming experimental variability parameters.

	Avg Success	Avg Epoch	Avg Errors
3NAND	83%	13.44	26.34
(A and B) or C	80%	14.81	25.11
A → (B → C)	88%	11.34	13.11
A nand B or C	82%	14.04	26.04
MAJ	84%	13.07	19.01
A and (B or C)	85%	12.94	22.95
3AND	82%	14.85	26.71

Table 2.8: Simulated results for Second Threshold (SR) Learning assuming experimental variability parameters.

encode positive and negative weights. All see a given example digit simultaneously, and all correct weights simultaneously as given by the error case (if there is one). This presentation and electrical scheme is the same as already presented for the OCR/Sci-kit digits scheme in Section 2.4.8, except for the fact that the system is now much larger due to the additional input dimensionality.

By the end of training, the algorithm has progressively adjusted weights (device conductances) such that each neural learning unit emulates a binary classifier between the chosen class and all others (geometrically, this solution is a hyperplane in p-dimensional space). As visible in Fig. 2.24(a), performance on this task is robust: top performances for SO and SR modes are at 86 % and 88 % respectively, and mild dispersions around the thresholds and $G_{\max}$ are not noticeably detrimental. While SR mode continues to slightly improve performance as it receives more training samples, SO performance markedly declines after a certain number are given due to a saturation of all devices towards $G_{\max}$. Similarly to the logic gate case, SR is superior in the symmetric case but is uniquely subject to poor performance due to device asymmetry (violent RESETs) effects. This effect is further assessed in Fig. 2.24(a)'s purple (uniform devices) and red (variable) series. These series suggest the case in which every RESET is a violent one: that, is, $\Delta G_{+} = 2.5\% G_{\text{Max}}$, $\Delta G_{-} = 5\% G_{\text{Max}}$. As visible, assuming all devices in the array suffer from this effect, the system would be incapable of resolving the MNIST task at greater then 70% accuracy even at full convergence (after sufficient examples have been presented).

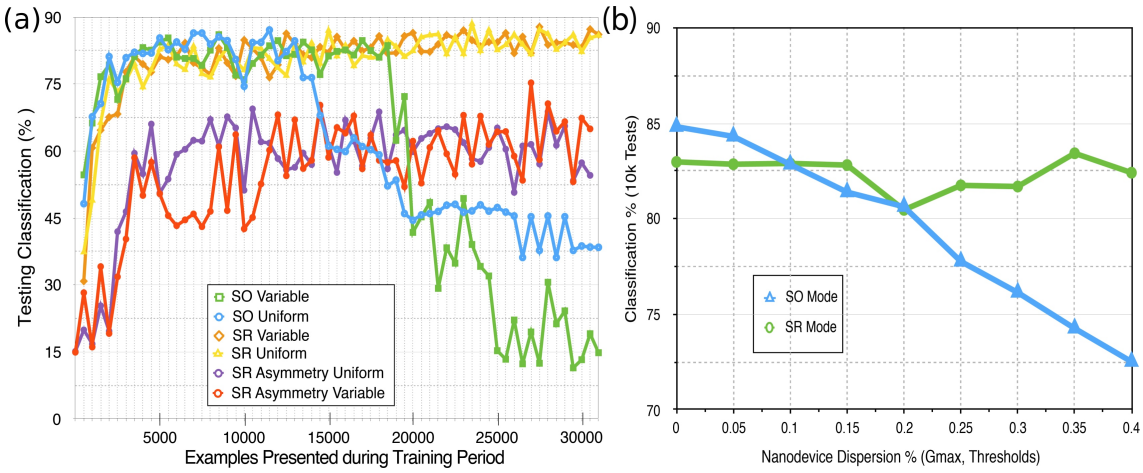


Figure 2.24: (a)Classification performance as percentage of 10,000 Tests on the MNIST database answered correctly (guess g matches actual class value k) as a function of the number of examples chosen randomly from the pool of training examples. Each point is the average of 5 simulations under different low starting conductances, in the uniform cases, and 10 simulations under different starting and dispersion values, in the variable cases.(b)Performance is given as the function of increasing dispersion parameter σ used to assign different threshold and maximum conductivity values to each of the simulated 15.6k organic memristive nanodevices for both SO, SR modes along a Gaussian spread. Every point is again averaged over 10 total runs.

On the other hand,Fig. 2.24(b) shows that systems with worse than experimental levels of variability continue to perform well on the task. SR is more resilient than SO as dispersions increase, which may relate to devices with very low $G_{\max}$ becoming 'stuck on' early (whereas they could be decreased in SR). The task shows that a large memristive perceptron using binary Widrow-Hoff approaches the natural accuracy limit of a perfect perceptron implemented in software (90 %). The higher performance on MNIST than the 3-XOR problem highlights a trade-off between exactness and efficiency in hardware ANNs. With logic functions every case of the truth table must be perfectly emulated, while in classification problems aggregate dynamics allows for approximately correct answers to emerge.

2.6 Discussion and Perspective

We explored the use of a unipolar organic memristor device with multiple thresholds, and demonstrated its advantages as well as limitations in both simulated and experimental supervised learning tasks. Our initial work formulated this learning in terms of the adaptive learning of logic functions (Sec. 2.3.4.1), and we found that on-chip learning is successful in this context. In addition, we found something special and unexpected, that the symmetry of our particular device's conductance evolution reduces the complexity of programming operations to one

pulse per cycle for a single function learned, and 2 per cycle in array. Moreover, the all-unipolar memristive latch scheme (Sec. 2.3.3.2) paved the way towards an all-organic on-chip design. This design was also confirmed to learn successfully in a non-ideal circuit environment.

Due to the simplicity of the model that yielded these results, we used a more complex analog compact model (Section 2.4.3.2), which was developed using concrete measurements of TBFe nanodevices. This model, notably allowed for a realistic temporal approximation of conductance evolution of the device, allowing us to appreciate which pulse-programming regime is most appropriate for learning in different contexts. In this context, we demonstrated interesting plasticity effects with the devices in the context of parallel neural learners architecture (Section 2.4.6), and subsequently adapted it to the full supervised case, where small handwritten digits were memorized and recognized on-chip (Section 2.4.8)

Lastly, our experimental work demonstrates the supervised learning ability of our organic memristive device in a physical, self-learing neuromorphic system; this is an important outcome that unites the two strands discussed in Section 1.4. In all of our past works, we always assumed the device would use both of its active conductance evolution modes, but in this work realistic physical constraints, in particular asymmetric RESET behavior, inspired us to consider a flexible programming scheme or learning style. In particular, both SET and RESET regimes can be exploited to effect suggested weight changes (second threshold/SR learning), or only SET (first threshold/SO learning). Examples of the learning of 3-input logic functions are shown for both styles and are compared to simulations.

Our physical system demonstrated high tolerance to inter-device variability in both first and second threshold styles (Section 2.5.4.3), which speaks to the power of our supervised learning algorithm. However, there is less resilience to imperfections involving the RESET mode. For this reason, while first threshold (SO) programming is a better choice for neuromorphic systems built with the present devices, improvement of devices for future use in SR mode represents a compelling future research direction. In particular, device engineering which improves RESET performance could unlock efficient pulse programming and thus considerable energy savings in larger systems.

Finally, through the use of a functional model directly inspired by the inter-device variability and asymmetry issues we encountered, (Section 2.5.5.1), we evaluated in greater depth how the architecture allows us to execute imperfect learning of non-linear function (in contrast to the experimentally demonstrated 'perfect' learning of linearly separable logic functions). In this context, our neural learner architecture accepts inputs more complex than a truth table and be chained together, as demonstrated also in the XOR learning example. This suggests that systems with many layers (crossbars), each composed of several parallel learners, can be built.

2.6.1 A bridge to multi-layer architectures

Multilayer memristive systems in principle should be able to solve canonical tasks with greater accuracy then we showed here due to their ability to generalize better on intrinsically non-

linear tasks. This is due to the difference between the computational capabilities of single feed-forward networks, which can only make linear approximations (as explained in Section 1.3.5,) and mulitlayer feedforward networks, which use a hidden layer allows to encode a higher-dimensional understanding of the considered task (see Sections 1.3.6 and 1.3.7). However, the physical implementation of multiple layer memristive architectures is an open topic of research, with many design choices and parameters unset or unconsidered. In the following topic, we consider this topic in great depth.

Chapter 3

Benchmarking Multilayer Memristive Learning Architectures

If a machine is expected to be infallible, it cannot also be intelligent.

Alan M. TURING

"THIS CHAPTER *studies larger on-chip architectures by expanding fundamental operations already demonstrated to larger multi-layer learning systems. These systems are evaluated on several data science (machine-learning) tasks. Here, more focus is on the systems-level operations and the performance of the designs than on the particular device they are built from, and the designs may be used with more than just the polymeric device from Chapter 2. However, typical failings of nanodevices are taken into account, and even in some cases, useful to the computing schemes being proposed.* "

THIS CHAPTER focuses upon the integration of memristive devices into larger learning systems. Special attention is paid throughout to the ease of realization and the resilience of the proposed schemes to problems or constraints with emerging nanodevices. There are four sections:

1. Motivation for exploring the design space of architectures for multi-layer online learning system is given, the considered benchmarking tasks are explained, and a generic, highly parameterizable nanosynapse model is introduced which is used in the context of the extended learning systems. (3.1).

2. Presentation of one layer regression, on-chip progression/regression learning system (ELM/No-Prop), and memristive MLP design (gradient-learning system) architectures that will be evaluated, (3.2))

3. Performance of each of these architectures on the benchmarks is noted, with a particular focus given on the MNIST benchmark as the primary one (3.3))

4. Critical constraints in terms of system size/overhead, nanosynapse richness, resilience to standard non-ideal effects such as variability and asymmetry, and energy constraints/requirements are discussed. (3.4))

3.1 Benchmarking Online Memristive ANNs

3.1.1 Research Question

Overall, in this chapter we extend the project of building scalable neurosynaptic learning system based on parallel and/or cascaded on-chip neural learner systems (as articulated in Section 1.4.5) to more powerful multi-layer systems that, in principle, should be able to generalize better on difficult tasks. However, there are many possible architecture configurations possible in the multi-layer learning context, which raises an important research question: which multi-layer architecture is preferable given realistic device constraints on candidate nanosynapses? To answer this question, we:

- Introduce a high-level choice possible between static, e.g. NoProp multi-layer learning systems, and multi-layer systems learning with backpropagation,

- Describe the data science tasks used to provide a constant basis of comparison,

- and detail the generic and highly parameterized nanosynapse model used in all of these systems to evaluate their resilience to key constraints.

3.1.1.1 Possible architectural tradeoffs

As introduced in Sections 1.3.5 and 1.3.6, Widrow-Hoff learning or the delta rule is also the engine of the iconic backpropagation algorithm [259]. Thus, in a multi-layer learning, a full backpropagation learning system is one option. In this context, on-chip friendly approximations of this architecutre have been detailed in simulated [180], experimental single-layer learning systems, [163, 181], and recently, a multi-layer prototype using this learning strategy was also realized [185]. Despite these successes, systems learning with the full backpropagation have two key constraints, one algorithmic and one related to co-integration with nanodevices: slow training/convergence, especially as system size (number of layers) and/or size of the training task is large [187], and deterioration relative to floating point performance due to imperfect device phenomenon (and, in particular, non-linearity device effects) [185, 256, 260].

A second general option involves an on-chip adaptation of the extreme learning machine (ELM) or No-Prop architecture. As described in Section 1.3.7, these systems use a combination of non-linear projections in first layer(s) and only trains the weights of the last layer [125, 126, 261]. An analogous approach of projecting to a higher dimensional space followed by linear regression has also been proposed in the neuromorphic engineering literature [128]. Moreover, this architecture possesses neuroinspiration, both to dendritic computation in individual pyramidal neurons [262], as well as to the non-linear encoding, linear-decoding neural population learning model called the Neural Engineering Framework (NEF) [100]. So far, ELM/No-Prop inspired projection systems constructed in part by nanosynapses have previously been described in [263, 264], but all assumed *ex-situ* learning to yield analytically derived second layer weights. We greatly expand upon these ideas and propose an ELM-inspired multi-layer learning system which can learn entirely on-chip (locally).

In the following sections we construct a direct comparison between the efficiency of both of these multi-layer (multiple crossbar) neural network schemes, with a control case represented by a single network learning system (crossbar) on a standard task in machine learning, the MNIST database of handwritten digits.

3.1.1.2 Considered Benchmark Tasks

To explore the computing possibilities of nanosynapse arrays, a few canonical tasks taken from the data-science literature were tested, with various input dimensionality $\boldsymbol{x}$ and task complexity c. As an analytical value for c is difficult to compute, we instead used an informal approach to estimate complexity by clustering dataset examples using t-Distributed Stochastic Neighbor Embedding (t-SNE), a well-known dimensionality reduction technique in the machine learning literature [255] and visually assessing the task's difficulty. The first task we considered is the Sci-Kit/OCR database of handwritten digits [254]. As introduced in the electrical simulations show in Section 2.4.8, this database consists of 1579 images of 10 handwritten classes (digits 0-9), each which has dimensionality $\boldsymbol{x_1} = 64$ (64 pixels). Some examples are visible in Fig. 3.1

(a). In addition, complexity c is relatively low for the task; as visible in the clustering plot (Fig. 3.1 (c)), the database could be easily classified as a combination of several linear functions due to the easily separated clusters (different colors in the plot). The second task we considered is a far larger and more complex version of the ten handwritten digits, the iconic M-NIST database first developed by Yann LeCun [151]. M-NIST consists of 60,000 training digits and 10,000 test digits; now, each example has a dimensionality of 784 pixels ($\boldsymbol{x_2} = 784$). A few examples from the M-NIST database are shown in Fig. 3.1 (b), which notably are more complex and irregular than those shown for SciKit. Moreover, the corresponding tSNE plot in Fig. 3.1 (d) demonstrates that clustering of these classes is notably non-separable or 'tangled' when mapped to a 2-dimensional space. As such, simple lines no longer suffice, and high-dimensional structures are required to yield strong classification results. Lastly, a very recent database called Fashion M-NIST was chosen which is the most difficult of all tasks considered [265], while being the same dimension as the M-NIST task ($\boldsymbol{x_2}$). Like the other two cases, there are ten classes to distinguish; these include shirts, blouses, and pants, a few of which are shown in Fig. 3.1 (c). Meanwhile, the corresponding tSNE clustering plot (Fig. 3.1 (f)) shows that the classes are even more entangled in the 2-dimensional space than M-NIST, a sign of very high complexity. Fashion M-NIST has the same total number of testing/training examples as M-NIST.

Although all three databases are benchmarked occasionally in the following results section, the MNIST database forms the bed-rock of analysis since benchmarks for the computing task are available within the field of neurmorphic engineering, making it possible to usefully compare the upper limits of our approaches with contrasting/competing neuromorphic approaches.

Like in standard computer vision tasks, images are unwrapped and presented as a single vector to our nanosynapse learning systems. To make them appropriate for a nanoelectronic environment, a mapping from analog pixel values to voltage values was made on a pixel-by-pixel basis within the range $-V_{\text{read}} < V_i < +V_{read}$ as

$$V_i = 2V_{\text{read}}(X_i / L_{\text{max}}) - V_{\text{read}}, \tag{3.1}$$

where L_{max} is the maximum pixel intensity, and V_{read} is a voltage that does not itself alter nanodevice conductances; critically, this allows for non-disruptive inference steps.

3.1.1.3 Fully Parameterizable Nanosynapse models

In order to achieve relevant benchmarking results for nanosynaptic learning systems, we needed to develop a generic nanodevice model (*i.e.*, applicable to designers of many species of memristive devices) that internally considers the implications of intra-device imperfections (nonlinearity, asymmetry) as well as inter-device mismatch.

These two important features could not be imported from the modeling work previously presented in Chapter 2, which developed analytical models either directly (Section 2.4), or in-

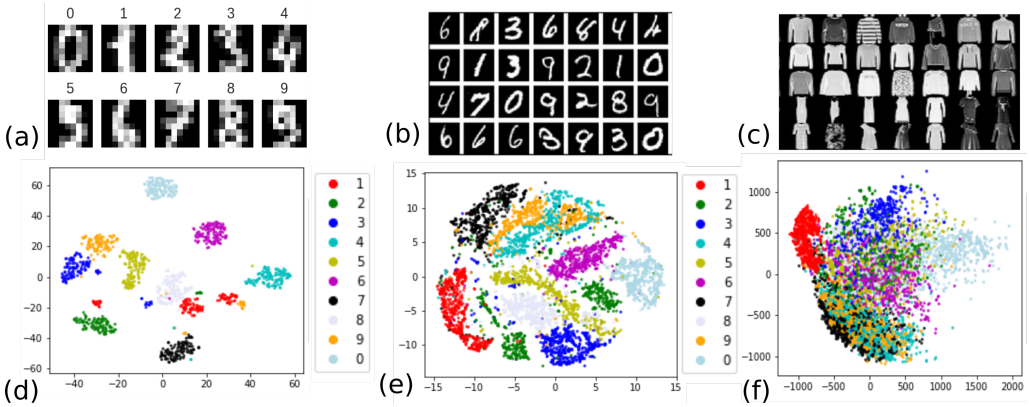


Figure 3.1: (a), (b), (c): Demonstration of examples from the SciKit Digits, M-NIST, and Fashion-MNIST databases, respectivel. (d), (e), (f): Demonstration of an unsupervised clustering of the entire database of SciKit Digits, and 5000 samples from the MNIST and Fashion-MNIST databases, using the Barnes-Hutt implementation of t-distributed Stochastic Neighbor Embedding (tSNE) method [255]. In all graphs, the axes are unit-less, the colors correspond to the labels or classes, and the colors/legend is the same among all graphs.

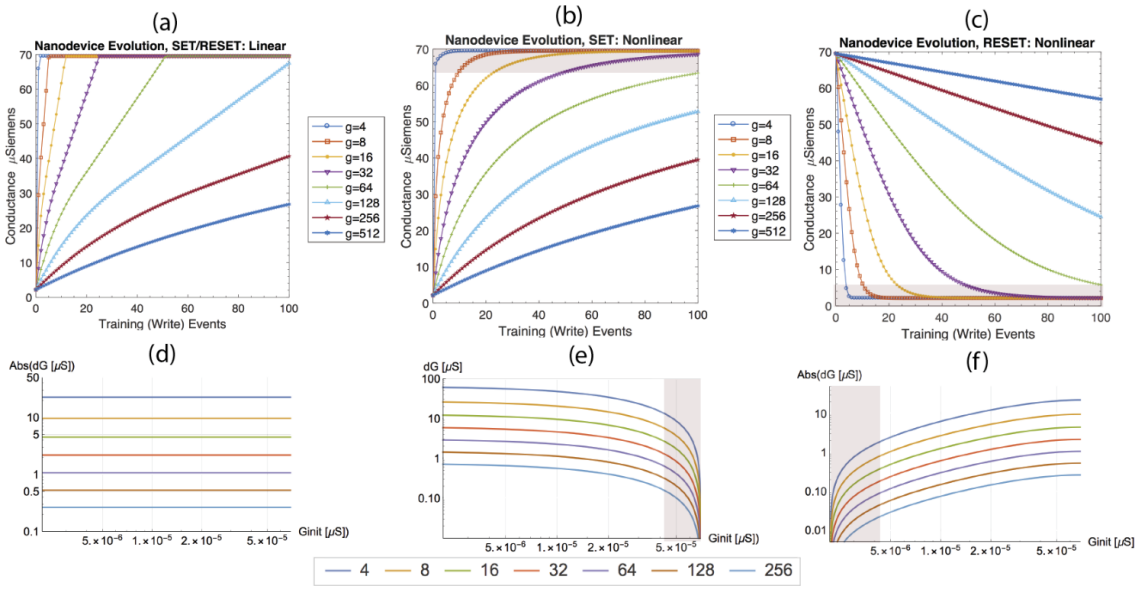


Figure 3.2: (a),(d) demonstrate symmetric linear model for both SET/RESET in which conductance evolves only as a function of the device's available weight space ($G_{max} - G_{min}$) and number (g) of writable levels; (b), (e), and (c),(f) demonstrate non-linear model in which positive (SET) and negative (RESET) conductance evolution, respectively, are also dependent on the starting conductance (G_{init}). Plots (a)-(c) share the same legend, are on a linear scale; (d)-(f) also share legend and are on log-log scale.

directly (Section 2.5.5.1) inspired by the physics of our particular polymeric device, and at least in the first case, was always uniformly applied for every device in a learning array. Nevertheless, our characterization of physical devices was a major guidepost; it suggested that behavioral

models must account for physical constraints in order to meaningfully predict behavior, and provided us with concrete demonstrations of these constraints. As discussed in Section 2.5.4.3, these constraints are many and had differential effects on learning outcomes.

Based on these observations, we have generalized the two models described in 2.5.5.1, and developed a third model which accounts for intrinsic device nonlinearity, as previously discussed in [260, 266]. This suite of three mathematical models: 'perfect", "variable", and "nonlinear", deal with these constraints at increasing levels of realism.

The first two models are based on identical equations, illustrated in Fig. 3.2(a) and Fig. 3.2(d). Conductance G evolves by a fixed amount whenever the device experiences a programming operation: it is changed by

$$\Delta G_L = \frac{r}{g-1}. \tag{3.2}$$

when a "positive" (or SET) programming operation is performed, and by $-\Delta G_L$ when a "negative" (or RESET) programming operation is performed ($r = G_{\max} - G_{\min}$; g, number of writable levels, is a now also taken as input to the model.

In the perfect model, all nanosynapses have identical $G_{\min}$ and $G_{\max}$ values; conversely, the imperfect model individualizes each device with different maximal and minimal values. As each device possesses a slightly different device range, each changes differently as a result.

These models are symmetric between SET and RESET behavior and linear as a function of number of programming operations, which is often not the case with physical memristive nanodevices: SET pulses are more effective when the device is low conductance and conversely RESET pulses are more effective when the device is high conductance [170]. Indeed, our experimental results with the polymeric devices demonstrated that devices using SET repeatedly (SO mode) struggled to complete their learning, as discussed in Section 2.5.4.3

To account for this reality, we make use of a third model, illustrated in Fig. 1(b-c) and Fig. 1(e-f), which is both non-linear and state-dependent. Conductance evolution now follows

$$\Delta G_{NL} = \beta(G, r, g) exp(\frac{-G}{r}), \tag{3.3}$$

with β uniquely defined for each SET and RESET:

$$\beta_{\text{SET}} = \alpha_{SET}(G_{\max} - G)\Delta G_L \tag{3.4}$$

$$\beta_{\text{RESET}} = \alpha_{RESET}(G - G_{\min})\Delta G_L, \tag{3.5}$$

where α_{SET} and α_{RESET} are parameters of the model.

In such a non-linear model, many device states tend to cluster near the conductance extrema of the device. This is visible in Fig. 3.2 (b)-(c), (e)-(f) as the red (strongly non-linear) zones. To form a fair basis of comparison between the different models of nanosynapse, α_{SET} and α_{RESET} constants were used to fit the curve with far more than g discrete states along the entire space, but g writable levels within the remaining, quasi-linear weight space (clipped

within 10% of G_{max} in either direction).

3.2 Single and multi-layer on-chip learning

3.2.1 On-Chip Regression Scheme

Before exploring the dueling multi-layer architectures, we need a stonger foundation for understanding learning in the one-layer (single crossbar) case. In this section, we first generalize upon the designs already shown in Chapter 2 Sections 2.4.8, 2.5.6.

Since neural networks require negative weights, and conductances are physically positive values, we always associate nanosynapses in pairs of differentially accessed devices. In fact, two basic topologies are possible for such a system. If a neural network layer has N inputs and K outputs, the "dual input" topology has $2N$ inputs and K outputs (Fig. 3.3 (a),(c)), while the "dual output" one has N inputs and $2K$ outputs (Fig. 3.3 (b),(d)). Electrically, in the dual output situation, input is applied as a set of voltage-mode inputs $X_1 \cdots X_{n+1}$; in dual-input, each input X_i is biased via sign-symmetric wires (X_{i+}, X_{i-}).

Output current values Y_k are obtained naturally through Kirchhoff's laws. For the dual output case this is done by taking the difference of the two output lines,

$$Y_k = \sum_{i=1}^{N} W_{i+,k} X_i - \sum_{i=1}^{N} W_{i-,k} X_i. \tag{3.6}$$

where $W_{i+,k}$ and $W_{i-,k}$ are physically analog conductance values ($G_{i+,k}$,$G_{i-,k}$) (the bias can classically be included by including a constant input).

For the dual input case, the output is naturally obtained on a unique wire:

$$Y_k = \sum_{i=1}^{2N} W_{i_{+/-},k} X_{i_{+/-}}. \tag{3.7}$$

During training stages, the final output of each neuron, O_k, is a logic signal representing its sign (-1, 1). This can be obtained using a simple circuit with a transimpedance amplifier, as described in Section 2.5.4 as well as [220]. When compared to Y_k, the appropriate error case, if any, is revealed. For ANN applications, during testing or inference stages, the sign-based output is no longer appropriate, and analog values amongst all output neurons of the ultimate layer must be compared. To realize this, the following function must be implemented on-chip, as was described conceptually in Section 1.3.5:

$$y_* = argmax(\boldsymbol{y}) \tag{3.8}$$

Where $\boldsymbol{y}$ is the analog vector of all outputs from the neurons. This approach, also referred to as max-out or winner take all, is a staple of learning operations in neuro-computing [267].

Online learning in such a physical neural network takes place by repeatedly alternating between inference and programming modes. In the former, training images are applied and the system output immediately demonstrates the error cases; in the latter, appropriate programming pulses are applied. Simple programming pulses can naturally implement learning. The precise voltage levels and number of steps needed to properly implement learning depends on the chosen nanodevice. Examples for every possible conductance evolution variety are shown in [245], while cases for the common bipolar, one-threshold device as well as the two-threshold, unipolar device inspired by the TBFe are visible in Figure 3.4, (iii)-(iv). Notably bipolar memristive devices require dual pulses per programming step, while unipolar memristive devices require only one. The combination of post-synaptic programming voltage pulses V_p and pre-synaptic, V_n, reduce error within all pair weights simultaneously (Fig. 3.3 (c),(d)), satisfying a simple version of the Widrow-Hoff learning algorithm : [118]:

$$\Delta W_{i,k} = \Delta G sign(X_i(T_k - O_k)), \tag{3.9}$$

where $\Delta W_{i,k}$ is the weight change at each programming step for the synapse connecting input i to output k, T_k is the expected output, O_k the actual output, X_i the input value, and ΔG the conductance change given by the device model.

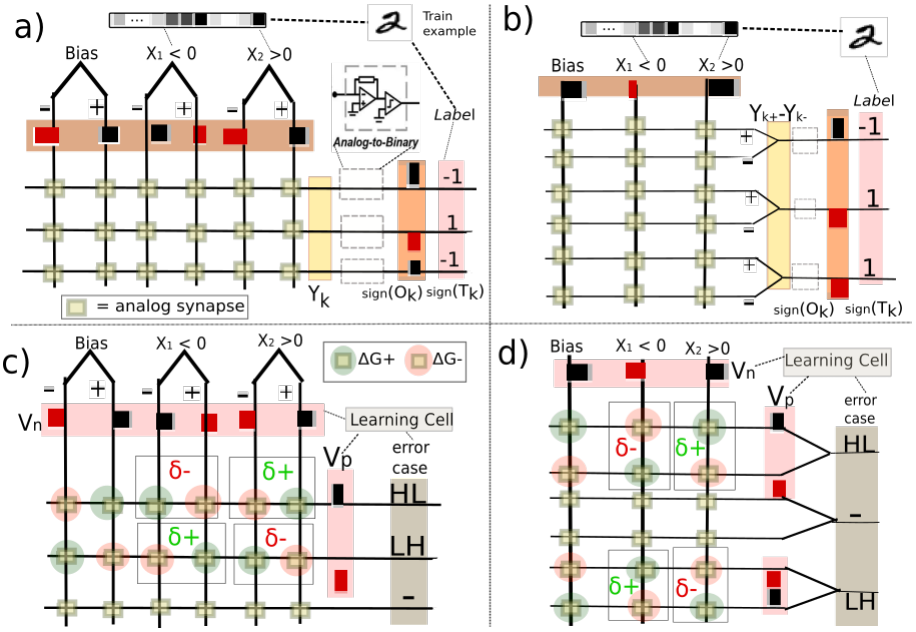


Figure 3.3: (a) Dual input system during inference mode (b) Dual output system during inference mode (c) Dual input system during an error-correcting (programming) step; the first two lines are active, while the third remains in non-volatile state. (d) Dual output system during an error-correcting step; now, only first, third output neurons are programmed. Outlined synapses pairs have their total weights changed by either δ_+ or δ_-; to realize weight pair changes, individual devices evolve positively/ are SET (green) or negatively/ are RESET (red). respectively.

3.2.2 On-Chip Multi-Layer Random Projection (NoProp) system

The NoProp uses a two-layer neural network pairing a crossbar of random weights with a crossbar of trained ones, as conceptually introduced in Section 1.3.7. The first layer's weights W_{in} are random, and are used to non-linearly transform inputs: the state of the jth hidden neuron (of M total) is given by:

$$H_j = f(\sum_{i=1}^{N} W_{in,ij} X_i), \tag{3.10}$$

where f is a non-linear activation function. In hardware, the random weights can be physically realized by using the natural dispersion of conductances around G_{on} and G_{off}. As noted in [264], this can be realized by turning all devices on or off. For our simulations, we always used the dispersion around the G_{off} state, using measurements extracted from the TBFe devices that we investigated in 2.2; experimentally obtained G_{ON} and G_{OFF} dispersions are shown in Fig. 3.4 (i). Projections from this first layer are then transformed from analog currents to voltages. Here, we used the simplest activation function to implement: $f(x) = sign(x)$.

In software/offline learning, the weights of the second layer W_{out} are computed using a Moore-Penrose pseudo-inverse, or a regularized form of ridge regression. These schemes involve complicated arithmetics and thus would imply considerable overhead on-chip.

Here, we propose a much simpler scheme that approximates this solution incrementally, *i.e.*, gradually writes the optimal W_{out} on-chip. To realize this, we use a crossbar with weights W_{out} (Fig. 3.4), which implements the second-layer of the neural network. When W_{out} is biased, again a post-synaptic currents (Y_k) is obtained and converted into voltage outputs O_k. When compared to expected T_k, on-chip regression is implemented using digital Widrow-Hoff, using the conditional pulse programming scheme detailed in Section 2.5.2, and shown in Fig. 3.4 (iii) . Three hidden-layer variations, presented in Fig. 3.4(b), were considered to implement a full range of weights:

- $M+1, s$ (1 extra line, simple): a single row of devices acts as negative reference for all other neurons;
- $M+1, c$ (1 extra line, complex): each row (n) is subtracted from the one following ($n+1$), first from an extra row (note that this requires sequential rather than instantaneous read-out);
- $2M$ (double lines): pairs of projection lines are used, exactly as in the dual output design (notes this doubles the area overhead required for the projection crossbar)

These three options are visually contrasted in the inset (ii) of Fig. 3.4, where the red line shows the constant and/or variable line which is being used as he reference line. Note that in the second case, the reference line is dynamically changing through time, and thus read-out from the first layer is not instantaneous.

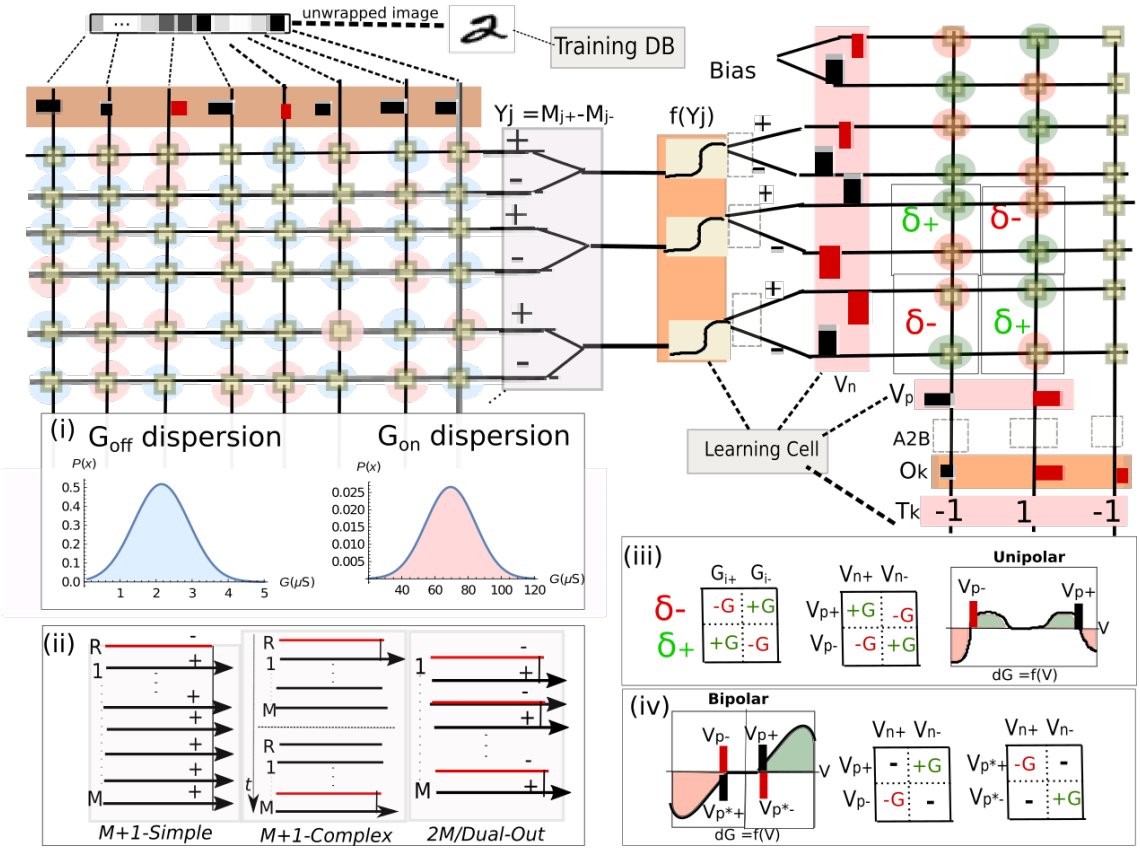


Figure 3.4: Main diagram depicts a NoProp system in the midst of a single training step. The first layer (W_{in}) is in either OFF or ON state (perfect model), or experimentally derived dispersions used automatically when imperfect and non-linear nanosynapses are chosen; Inset (i) shows examples of nanodevice variability used to naturally obtain random first layer weights, where the values are experimentally derived from the TBFE devices characterized in Chapter 2. Next, projection to the second layer is realize using schemes one of the schemes given in inset (ii); 2M is shown in the main diagram. Finally, second layer weights (W_{out}) are corrected in accordance to the error cases shown using pictured conditional pulse programming scheme. Inset (iii) shows the general delta rule implementation and how the unipolar TBFe devices can implement learning in one parallel learning step. In contrast, Inset (iv) shows the case for generic bipolar memristive devices where dual programming pulses of opposite polarity are required to fully implement the delta rule.

3.2.3 On-Chip Multi-layer Gradient Learning system

Now, we contrast this nanoelectronic vision with another alternative for learning in a multiple-crossbar environment. As conceptual introduced in Section 1.3.6, a full multilayer perceptron trained by backpropagation must also iteratively train the input weight matrix, W_{in}. In this work, we build upon the conceptual insights provided in Section 1.3.6.2 to demonstrate an on-chip MLP system learning with a cross-entropy (log-loss) cost function, and softmax one-hot encoded outputs (*e.g.*, binary labels; target classes are 1 and non-target classes are 0). This

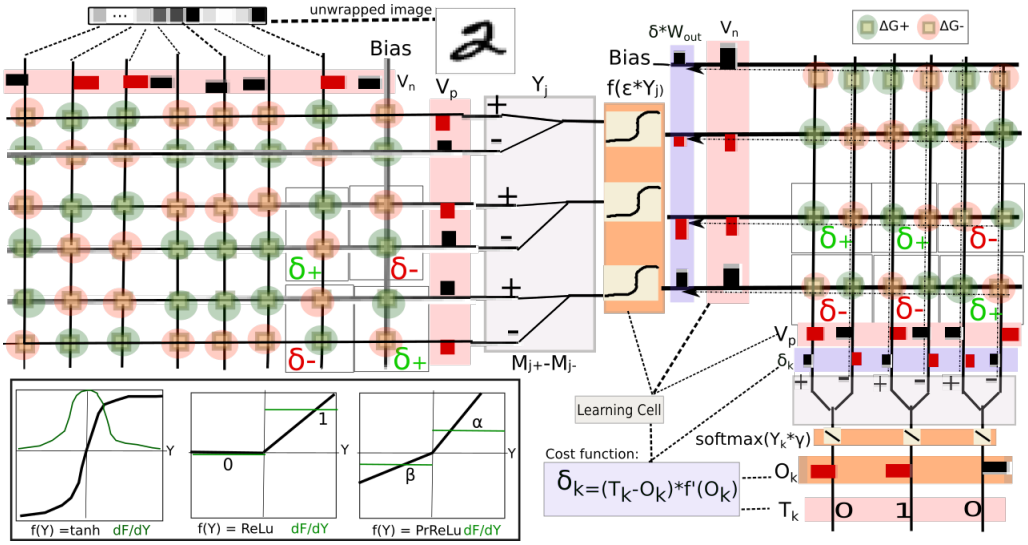


Figure 3.5: A gradient learning system is depicted during programming step for the single example (SGD) system. In this case, a single conditional programming step may be used to improve all error cases in all layers. A few possible activation functions, and their derivatives, are visible in the inset.

choice indeed leads to excellent performance in terms of machine learning, and maps relatively naturally to nanodevices crossbar. Softmax output means that the circuit output O_k is obtained as

$$O_k = \frac{\exp^{\gamma Y_k}}{\sum \exp^{\gamma Y_k}}, \tag{3.11}$$

where Y_k is obtained as in Eq. 3.6, and γ is a normalizing parameter. Softmax output is a significant cost in terms of circuit overhead. It could be implemented by op-amp circuitry; however, as each output neuron's computation depends on the others, this requires calculation outside the crossbar arrays and increases computations per training cycle. Nevertheless, the choice of softmax leads to learning rules that are simplified with regards to apparently simpler output choices. Indeed, following standard backprogation calculations, the weights in the output layer should be adapted as:

$$\Delta w_{j,k} = -\eta \delta_{j,k}, \tag{3.12}$$

where η is a learning rate, and $\delta_{j,k} = H_j(O_k - T_k)$. For the input layer,

$$\Delta w_{i,j} = -\eta \sum_k \delta_{j,k} w_{j,k} f'(H_j), \tag{3.13}$$

where H_j is the activation of that middle layer neuron, and $f'(H_j)$ its derivative. See Section 1.3.6 again for the full derivations of these equations.

A critical constraint in on-chip backpropagation is that activation functions must be differentiable; the $sign$ function used in the ELM-inspired system just introduced in Section 3.2.2

does not meet this criteria, as its derivative is 0 almost everywhere, and even more complex stochastic or quantization amendments to it have mixed efficacy [268]. Therefore, we introduce two additional activation functions: a $tanh$ activation function, and a rectifying linear one. In the rectifying linear family, which has achieved state of the art results in machine learning [153], positive outputs are projected by constant α, hence $\frac{dH_j}{dx} = \alpha$, and negative ones by a constant β, thus $\frac{dH_j}{dx} = \beta$. We have considered and contrasted both options in this work. Notably, while the derivative $f'(H_j) = 1 - tanh(x)^2$ varies analogically, rectifier's gradients are constants; this significantly reduces constraints on accompanying analog CMOS circuitry.

The sum in equation 3.13 used to propagate error backward, can be performed *in-situ* using the dot product operation [269]. Nevertheless, the calculation of gradients for every pair of nanosynapses and its on-chip implementation is non-trivial, and a considerable circuit overhead with regards to the NoProp system. In this work, to make the backpropgation easier to implement with memristive devices, we simplify this rule to a sign-based implementation appropriate for on-chip learning. This is achieved by adjusting weights according to $(sign(\delta_k x_{j,k}))$ in both layers [218]. This has the added advantage of transforming stored gradients from analog values to binary ones. For ultimate energy efficiency, these binary gradient values (error-cases) could be stored locally for automatic programming by the on-chip memristive latch scheme already demonstrated in Section 2.3.3.1.

We consider two variants of this on-chip learning scheme; in the single example stochastic gradient descent (se-SGD) implementation, programming steps are made after every example image as in the two previous systems; in mini-batch stochastic gradient descent (mb-SGD), error is accumulated over a set of n training samples before a programming step. There are significant differences between the two approaches in terms of over-head and programming costs. After each presented example in stochastic training, the vector of post-synaptic errors $\boldsymbol{e}$ and the vector of pre-synaptic values $\boldsymbol{x}$ stored as binary values can easily be stored at the periphery of every row/column. This means that se-SGD requires no more special CMOS programming accessory than the previously demonstrated programming for the NoProp system, and can rectify all error-cases on-chip with the single conditional programming step (for unipolar devices), or with dual conditional programming steps (for bipolar devices).

However, during batch learning or mb-SGD policy , this favorable periphery property no longer holds, since gradient descent is always manifested with regards to the unique pre-synaptic and post-synaptic combination of error at every synapse in the array, as given by Eqns. 3.12, 3.13. As demonstrated in Fig. 3.6, far more than a single simultaneous programming step may be required; in the worst/standard case, as many as there are columns of devices. As also discussed in [181], this accumulation operations within every batch additionally requires non-negligible hardware overhead. Specifically, mb-SGD would require associated memory devices to continually store/add error gradients. However, by storing only the sign (binary values), this complexity is greatly reduced compared to a fully analog proposal.

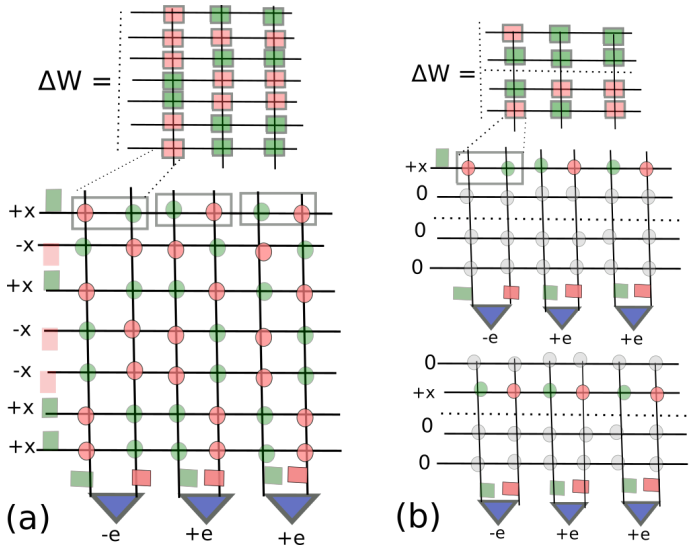


Figure 3.6: (a) se-SGD; top shows the algorithm defined ΔW for weights, and bottom its crossbar writing scheme, where here differential weights are expressed relative to double output weights (gray boxes denote equivalence of two devices to a single nanosynapse). As visible, all of the writing pulses can happen simultaneously (in one writing pulse). (b) mb-SGD; top shows accumulated gradient for every weight in the matrix may no longer correspond necessarily to a periphery-compatible writing scheme. Thus, as shown in two progressive time frames in the bottom pane, mb-SGD error correction scheme may take place in a line-by-line writing style in the crossbar array. This may require as much as nxm associated memory to store the gradients for each nanosynapse (device pair).

3.3 Performances of all considered systems

3.3.1 Performance of standard one-layer scheme

First, we benchmark the online learning capabilities of the one-layer (single crossbar) nanosynapse learning systems. Figure 3.7 (a) shows that both dual input/ dual output designs successfully converge on the MNIST task at between 85-88 % recognition rate, with devices with number of states $g = 256$. Depending on the architecture choice, online learning systems using the Widrow-Hoff rule lose either negligibly (dual-input; red) or 2-3% in performance relative to the *ex-situ* or imported weights control case (89 % on the test set). As visible, *in-situ* performance always slightly trails the *ex-situ* approach, yet by two training epochs (120,000 samples), *some* of the considered nanosynapse models/designs have converged to this level. Notably, dual input systems (red) slightly outperform dual output (blue). While both linear (perfect, imperfect) models achieve roughly comparable performance, non-linear models lose significantly: 6-7% in the dual input case, and 8-9 % in the dual output case.

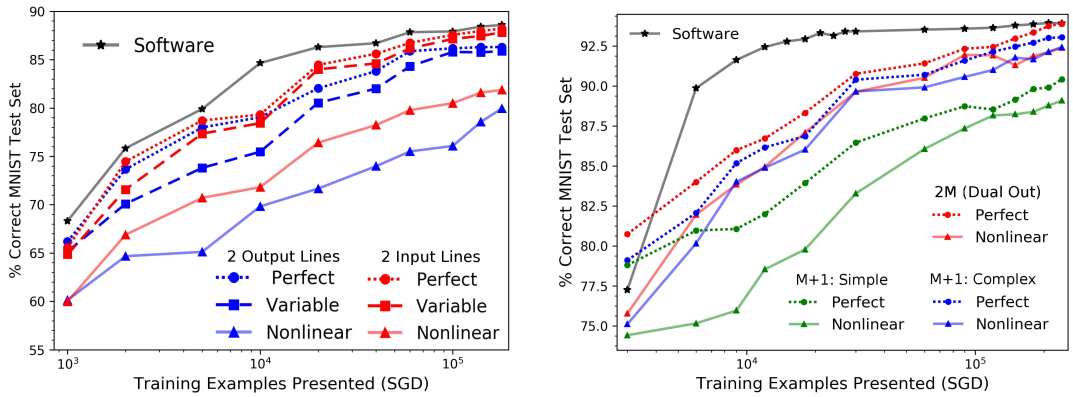


Figure 3.7: (a): One Layer system convergence as a function of training samples presented. (b): NoProp convergence also as a function of samples presented, with both on-chip and off-chip solutions contrasted. Every system learns with nanosynapses with $g = 256$ or 8-bit resolution analog devices.

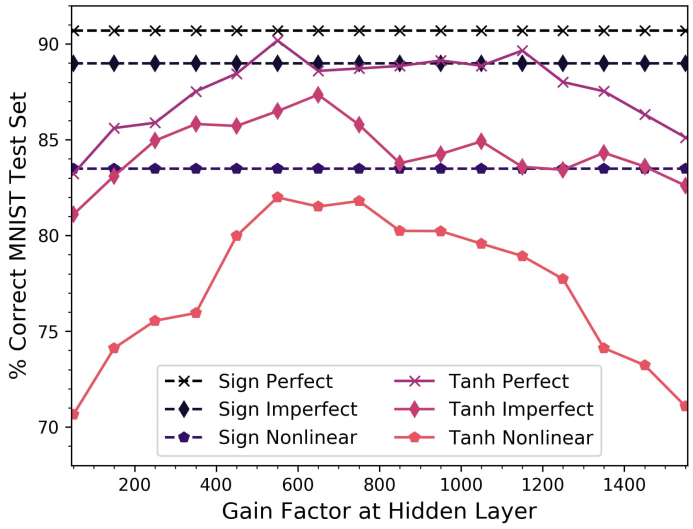


Figure 3.8: For two-layer learning systems using a random first layer (crossbar) and a second crossbar using Binary Widrow-Hoff, an analog gain factor ϵ is varied (where $tanh(x\epsilon)$,x being the pre-synaptic input from the first layer) for the three varieties of nano-synapses. These three series are contrasted to the three constant series using the digital, $sign$ activation function. Every point is the average of 5 simulations with new starting conductances in both layers.

3.3.2 Performance of random multi-layer scheme

Figure 3.7 (b) shows the convergence of all *in-situ* NoProp systems with the 3 considered projection schemes, alongside results obtained using computed output weights (ridge regression).

As visible, the *ex-situ* learning approach requires less training examples to obtain the optimal solution; when ($M = 3L$), maximum 93.8% performance is analytically obtained in around 30,000 samples; in contrast, a full 3 epochs or 180,000 training samples are required to achieve *in-situ* convergence. Of the three considered weight modes, only $2M$ projection with linear devices converges closely to the software solution; linear devices using $M+1, c$ projection scheme lose only 0.3 % of the off-chip solution, and non-linear devices using these models lose around 2 % in accuracy on the test set. Notably, the $M+1, c$ scheme does very poor overall, achieving 90.1 % (linear) and 87.9 % (non-linear).

Nevertheless, all NoProp systems do better with non-linear device models relative to one-layer systems; predictability (separability) of inputs is greatly improved once passing through the hidden layer as compared to direct presentation. With regards to relative success of the three projection schemes, it is worth nothing that dot-product current differences may need to be optimized with respect to the hidden layer neurons function. In the present scheme, a Gaussian probability distribution centered as close as possible to 0 optimizes the symmetry of the $sign$ activation function. The $2M$ /pair projection method reliably insures symmetry; the $M+1, c$ adequately approximates symmetry; $M+1, s$ method often results in asymmetric outcomes since it relies on a single row of devices as reference.

So far, all results with the random projection system have assumed a binary activation function at the hidden layer ($sign$), but we have also considered an analog alternative, the same $tanh$ function used otherwise in the gradient learning system. Figure. 3.8 shows that, far from being an advantage, an analog activation function in our studied system can be a disadvantage unless the correct value of the slope parameter ϵ, or the gain factor, is chosen. As demonstrated, at the optimal value between $\epsilon = 500$, the full performance of the digital/sign activation is emulated, but at too high or too low value, significant loss is incurred. This trend holds true for all three considered type of nano-synapses. This phenomenon has been earlier noted in [264], which commented on the critical role of the parameter. We note that, since there is no need for the activation functions in the progression-regression systems to be differentiable, the digital approximation seems unambiguously better for its purpose. The removal of another hyper-parameter can be a major asset to reduce the complexity of design and evaluation of NoProp or NoProp-inspired ANNs.

3.3.3 Performance of full gradient learning scheme

First, we used the proposed se-SGD learning policy (no minibatch), $tanh(\epsilon x)$ activation function, softmax output, cross-entropy error function, and $sign(\delta)$ to obtain the basic convergence results visible in Fig. 3.9 (a), green curves. When $g = 256$, systems with linear nanosynapses reach 87% and those with non-linear ones reach 83% on the MNIST test set. This is poor performance relative to system complexity, as the far simpler one-layer systems can achieve the same result. However, a direct comparison to software results obtained and imported as 8-bit weights to the crossbars shows that linear and non-linear devices lose only 1.5% and 4%

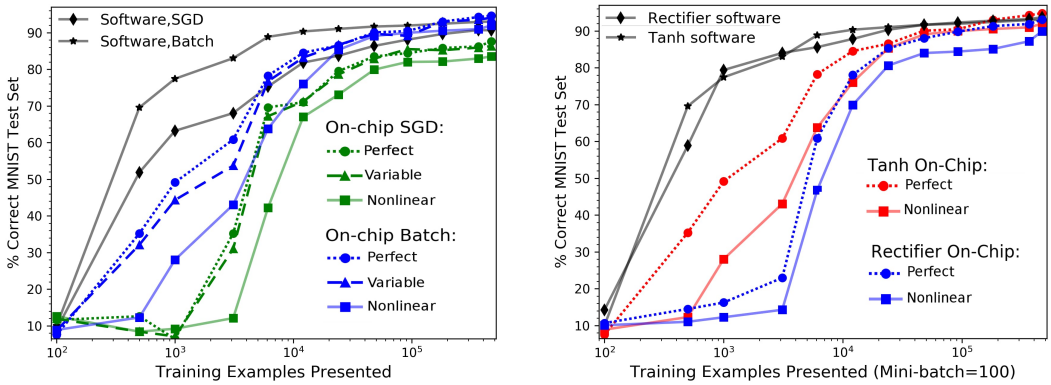


Figure 3.9: (a): Convergence for all $tanh$ activation function systems, including both se-SGD and mb-SGD approach. Mean performances ($n = 5$) over 10 epochs (600k Tr) are demonstrated when network is trained after every example. Here $g = 256$ addressable levels, $M = 300$, tanh is activation function. (b): convergence for all considered mini-batch systems. Mean performances ($n = 5$) over 10 epochs are again shown, with presentation in mini-batches ($b = 100$ each) and programming subsequently. $g = 256$ addressable levels; $M = 300$, tanh; $M = 800$, rectifiers.

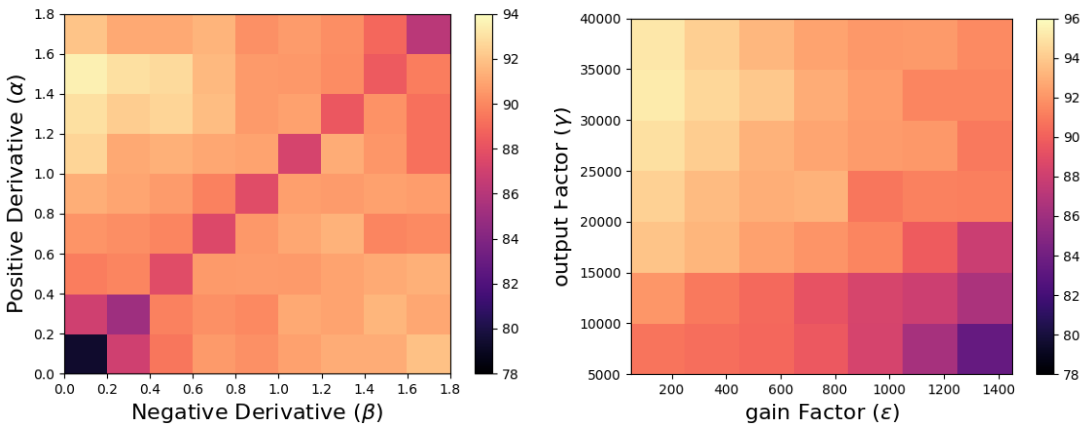


Figure 3.10: (a): Grid search of possible derivatives is performed for the various rectifier systems (b): Grid search of gain factor (ϵ) used in the inner layer of $tanh$ activation against the gain factor γ (used in the softmax projection of the ultimate layer). The colors correspond to the classification accuracy on the test set, as visualized in the colorbars besides each image.

from the maximum reachable, respectively. Evidently, multi-layer stochastic gradient both in software and hardware does not generalize very well on this dataset due to excessive updates. Moving from online or single example training to accumulated groups of examples can help ease convergence, since the order of individual samples in the groups does not matter.

To explore to which extent batch-learning should be applied, we tested progressively increasing the batch sizes from $b = 1$, which is the same as online/single-example, up to $b = 300$. Figure 3.11 shows that, in general, an optimal size for mb-SGD for both the fully analog ($tanh$) and digital ($ReLu$) hidden layer neurons exists around 80-150 samples. Thus, in all the following simulations we have always considered mb-SGD as $b = 100$.

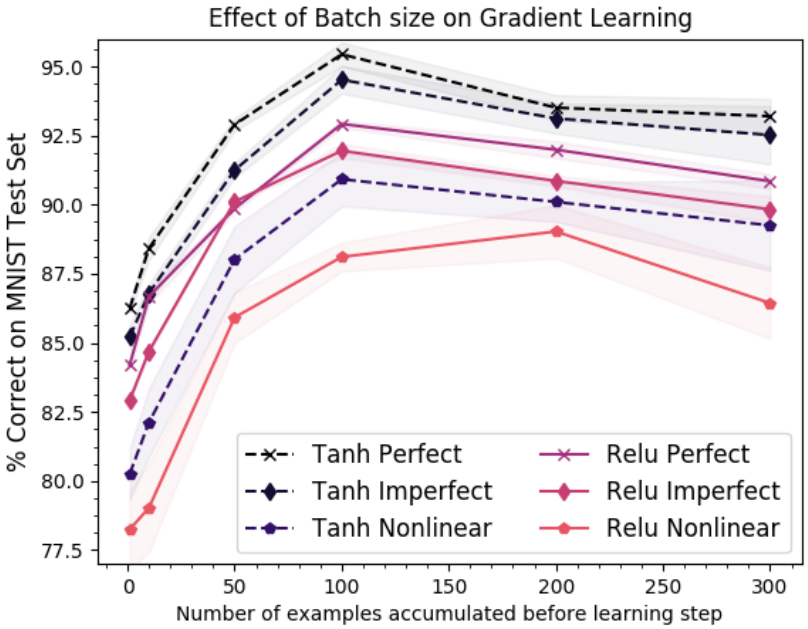


Figure 3.11: All six curves show the gradual transformation from the se-SGD to mb-SGD policy, for the three device cases given each hidden layer neuron considered. In all cases 7 epochs were given to converge, and every point is the average of 5 total trials with different initial random weights; dispersion from this mean is shaded.

Following this result, we proceeded to implement the mb-SGD learning policy with training images into mini-batches of $b = 100$ images and only applying training moments after this multiple-inference period. Again using cross-entropy and softmax error as the cost function and following sign-based gradient descent, Fig. 3.9 (b) demonstrates that the switch from SGD to batch learning improved performance on the test set to 96% for linear nanosynapse models and 92% for non-linear ones. The on-chip systems eventually reach the performance reached by software batch learning systems or even slightly outperform them, but only after 7-10 epochs (420k-600k training samples).

The effect of hidden layer activation function on performance is also assessed in Fig. 3.9 (b), which contrasts batch convergence using $tanh$ (already shown) to convergence of the best

performing rectifier system ($\alpha = 1.5$, $\beta = 0.1$). The former out-performs the latter, which also notably converges slowly. Final performance for $tanh$ systems is greater than 95%, and rectifier 93.5% in linear cases, and 92.5%, 91% in the non-linear cases, respectively (Table 1). However, stronger performance for $tanh(\epsilon O_j)$ does come at the expense of fine tuning of gain factor ϵ; as visible in Fig. 3.10 (b), generally lower gain factors perform better.

In contrast, rectifier cases are tuning free, as simple (linear) gradients are obtained automatically. Even if the optimal α, β values are not known *a priori*, choosing a standard rectifier such as ReLu ($\alpha = 1$, $\beta = 0$) still obtains reasonable performance; this well known function lost only 0.6% on average compared to the optimal rectifier as visible in Table 1. The possibility space of all α, β values is visible in Fig. 3.10 (a); this graph shows that, in general small constant negative derivatives and large positive ones achieve best results, while values along the diagonal do worse (when $\alpha = \beta$, the hidden layer function simplifies to a linear one, reducing the system's computational power substantially). Finally, Fig. 3.10 (b) shows that all systems using the chosen cost-function must use a large enough output factor γ at the softmax layer.

3.3.4 Comparison of all results on MNIST benchmark

With regards to the primary benchmark task, we have noted the performance of the best performing as well as the mean performance of 10 different systems for each of the considered architectures. As visible in Table 3.1, gradient learning systems with analog hidden layer functions and optimized parameters perform the best overall, achieving mean 96% on the MNIST. With still optimized parameters but digital hidden layer functions, performance drops 2%, mean of the optimal rectificer performance, and a 3% loss in performance when a standard, non-optimal digital neuron scheme is used (rectified linear unit). Interestingly, both of these performances are already slightly lower than the mean of the NoProp system, which can reach greater than 94% mean performance with a very large hidden layer. Finally, note that the $se-SGD$ gradient learning performance, mean 88% is a very disappointing result, as it only represents a small increase in improvement relative to the single crossbar learning environment ($85-86\%$).

3.3.5 Performances across all datasets

Finally, Table 3.2 shows the performance of all systems on all three tasks. As visible, the computational ability of models across all the tasks is a constant, with backprop in batch mode always achieving the strongest result, NoProp/ELM achieving the second strongest, and one-layer along with backprop se-SGD mode always performing worst and second-worst, respectively. However, this comparison also reveals the respective difficulty of the tasks. As visible, the Sci-Kit small digits/OCR task is by far the easiest, Fashion-MNIST the most difficult, and MNIST is an intermediate case closer in difficulty to the latter than the former.

Formally speaking, this difficulty relates to how non-linearly separable the classes are from

Nanodevice ANN Algorithm	Performance on MNIST Test Set [10k]		
	Mean Linear	Mean Nonlinear	Best
One Layer 2In	85.2%	84.1%	86.1%
One Layer 2Out	86.3%	84.9%	87.3%
NoProp SGD (M=3000)	94.5%	92.8%	95.2%
Backprop $tanh$ SGD (M=300)	88.3%	83.8%	88.9%
Backprop $tanh$ Batch (b=100)	96.1%	91.7%	96.9%
Backprop ReLu Batch (b=100)	93.2%	90.3%	93.9%
Backprop Rectifier Best, Batch (b=100)	93.9%	91.1%	94.8%

Table 3.1: Performances for all examined systems are directly contrasted on the same challenge (10k digits of the MNIST testing set), with mean values ($n = 10$) for the variable linear model and variable non-linear models shown as well as best system performance overall (always a linear model)

each other. As was previously visible in Fig. 3.1, in the small digits/OCR case the classes are visually quite linearly separable; for MNIST and Fashion-MNIST, these classes become increasingly difficult to distinguish. In the case of the Fashion-MNIST database in particular, exceeding 90% accuracy on the test set is not achievable with the present neural network set-up. Indeed, a single hidden-layer system (MLP) may fail to generate a sufficiently complex enough non-linear mapping function to separate these classes from each other. More complex gradient-learning architectures, e.g. the deep and/or convolutional network described in Section 1.3.9 would be required.

ANN Architecture	% correct on Respective dataset's Test Set (Average of 10)		
	SciKit Digits	MNIST Digits	Fashion-MNIST
One Layer	92.1	86.1	77.24
NoProp/ELM	97.7	94.2	84.92
Backprop se-SGD	93.4	87.5	80.22
Backprop mb-SGD	98.8	95.9	88.12

Table 3.2: All results are based on a characteristic convergence times of 2 epochs (single layer), 3 epochs (NoProp systems), and 8 epochs (gradient learning). NoProp Systems are M=3L and always with $sign$ activation; backprop systems M=300 and batch learning with $b = 100$; all backprop systems use $tanh$ activation. Every system learns using quasi-linear nanosynapses with 8 bit weight space ($g = 256$). Listed values are mean of 10 iterations with different initial weights and presentation of examples shifted throughout the epochs.

3.4 Critical constraints in considered learning systems

3.4.1 On the basis of Hidden Layer size

While single layer systems (dual-in, dual-out) have fixed sizes set by the dimension of the input L and the number of parallel perceptrons or classes N, the considered multi-layer NoProp and gradient learning systems have flexible hidden layer sizes M; varying this parameter has a strong influence on the quality of the learning outcomes. Fig. 3.12 (a) shows that only NoProp systems with larger hidden layers generalize well on the test set. Only when $M = 2L$ is benefit over the one-layer parallel perceptron system evident; indeed, at very small hidden layer sizes, performance is actually inferior to one-layer, direct task presentation. At dimensions greater than $3L$, the system improves towards 95% classification accuracy, in the case of linear devices, and 93.5% , for non-linear devices. Among the considered weighting schemes, $2M$ systems on average gain about 2%-3% in classification performance relative to $M+1$ complex, and non-linear systems lose, on average, 1%-1.5% relative to systems with more ideal linear models. As visible, only 2M systems with more ideal nanosynapses can emulate the results obtained using analytically-derived weights.

The quality of generalization of gradient learning systems also depends on M, although less dramatically. As visible in Fig. 3.12 (b), the $tanh$ function performs better with a smaller hidden layer size, between $M = 100$ and $M = 300$, whereas rectifier systems perform at their best between $M = 600$ to $M = 1000$. In the former case, too large systems suffer from over-fitting, as a number of over-complete neurons leads to confusion in learning outcomes. In the latter case, it is worth noting that rectifier hidden layers already implement a form of implicit "drop-out"- a form of normalization that battles this sort of over-fitting. Explicitly, synapses in the first layer connected to activations for 'dead' (negative) neurons - ReLu ($\beta = 0$), or asymmetrically small outputs ($\beta < \alpha$) do not adjust, or hardly adjust to weight updates, respectively.

3.4.1.1 On the basis of Nanosynapse richness

Figure 3.13 contrasts the performance of single and multiple crossbar to solve all three considered data science tasks (SciKit/OCR, M-NIST, Fashion-MNIST). As demonstrated in the contrast between the solid lines (one-layer/crossbar) and dashed lines (two-layer/crossbars), a hidden or projection layer not only provides a decisive advantage in classification performance, but also allows less rich nanosynapses to perform online learning than in the single crossbar case. In the graph, this is visualized as the leftward movement of the corresponding data science curves. While all systems fail to learn with binary devices, the difference is especially profound from 2 bits ($g = 4$) to 5 bits ($g = 64$); at 7 bit ($g = 128$) and greater resolution, differential effect of nanosynapse quality is less.

Next, we have extended this analysis to conduct a direct comparison of all three learning systems, including multi-layer gradient systems, and to consider the impact of device non-

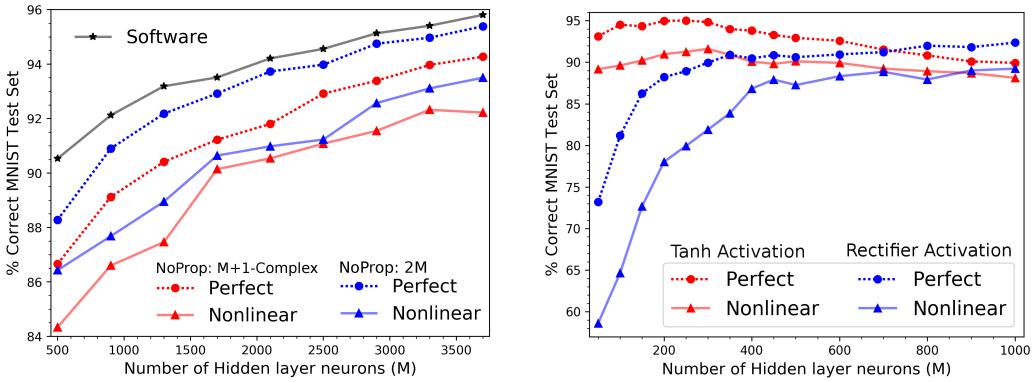


Figure 3.12: (a): Response of the two best NoProp designs to increasing hidden layer M. Mean of 5 simulations, with $g = 256$ addressable levels, and 3Tr epochs given for convergence. (b): Response of gradient learning systems, using $tanh$ and best performing rectifier activations and mini-batch of $b = 100$, to variance in M. Mean of 5 simulations, with $g = 256$ addressable levels, 7Tr epochs given for convergence.

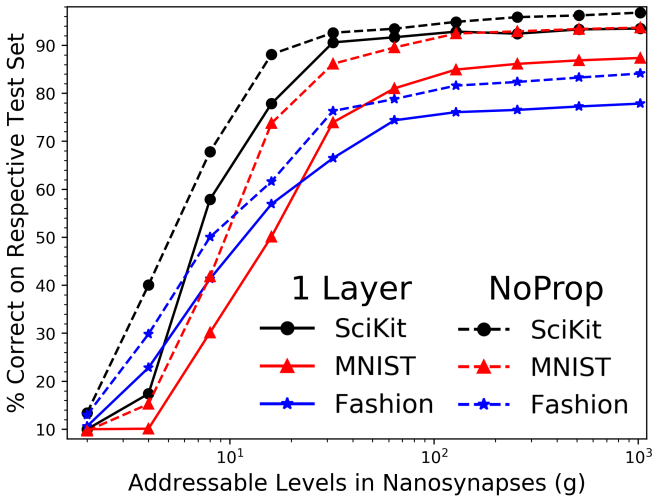


Figure 3.13: All 3 data science tasks are attempted with systems built from increasingly deep/rich nanosynapses, within the range of 1 bit ($g = 2$) to 10 bits ($g = 1024$). One layer systems are a single crossbar of double dimensionality of the task as input and the number of classes as output. For 2-layer or double crossbar learning systems, the hidden layer size is $M = 3L$; that is, $M = 2352$ for MNIST and Fashion-MNIST, and $M = 192$ for the simple SciKit database. In every case the linear/ideal model is assumed for the nanosynapses in these learning systems

linearity. Figure 3.14 shows the dependence of all considered learning systems upon nanodevice analog richness given the linear variable model, and the non-linear model, respectively, for the M-NIST database case. Notably, all considered learning systems perform very poorly below 32 addressable levels/5 bits, for the linear models, and 64 levels/6 bits, for non-linear ones.

While at 6 bits the single crossbar and NoProp systems are both within 1-2% of their maximum achievable test-set performances, even given non-linear device behavior, 7 bits ($g = 128$) are required for linear gradient learning systems to do the same, and 8 bit ($g = 256$) is required for gradient systems built with non-linear devices (the most realistic case). This is a very significant contrast which is explained mostly by the varying requirements of the two learning algorithms. While Widrow-Hoff only needs a hyperplane to separate classes, gradient systems traverse a more complex error function/landscape.

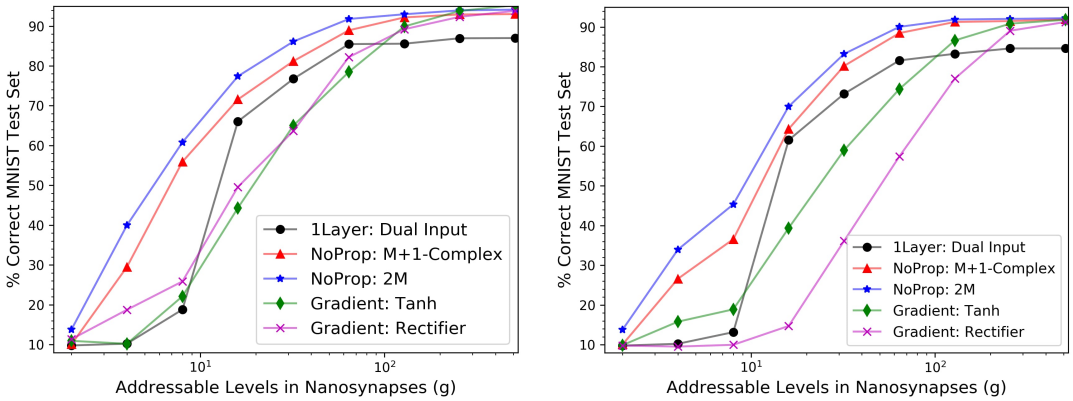
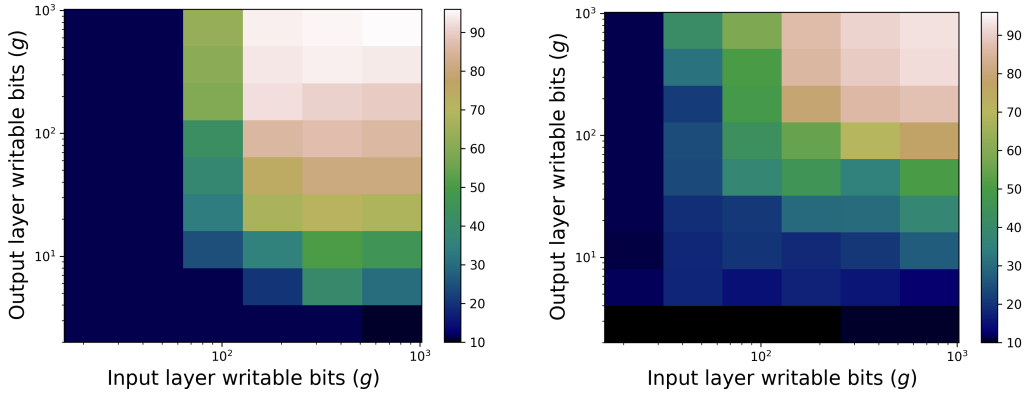


Figure 3.14: (a): Effect of nanosynapse richness for all considered learning systems using linear model. (b): Effect of nanosynapse richness for all considered learning systems using non-linear models.
All regression (one and two layer) systems are given three epochs , and gradient systems are given seven epochs, to converge. Mini-batch style programming was used for all gradient learning simulation cases. $M = 300$ for gradient Tanh; $M = 800$ for gradient rectifier; and $M = 3000$ for no-prop. Every point is average of 5 runs.

3.4.2 Differential constraints on nanosynapse depth

As discussed before, the same quality of nanosynapse is used throughout the system. In the case of the NoProp system this comparison does not matter because the first layer weights are not trained; however, in gradient learning systems where both layers are trained, nanosynapses with different depth may be used in different layers. Given a differential nanosynpase depth between the layers, where g_1 is and nanosynapse depth in the first/input layer depth and g_2 is the depth on the output layer, a search of the parameter space was conducted in order to discover additional constraints on learning. As visible in Figure 3.15 for the MNIST task, gradient learning systems learning with both layers in batch-mode are notably more sensitive to the depth of the first layer weights (x-axis) than the second layer weights (y-axis). For instance, while between 3-6 bits in the second layer can still achieve between 50-90 % performance when first layer has 8 bits or more (bottom right grid search), the inverse case (top left grid seach) achieves

little better than chance. The contrast between Fig. 3.15(a),(b) shows that while non-linear devices reduce performance overall, they may slightly reduce this stringency requirement on the first layer devices.



(a) Nanosynapse depth is differentially varied in two layer learning systems with linear synapses

(b) Nanosynapse depth is differentially varied in two layer learning systems with non-linear synapses

Figure 3.15: Both systems depict gradient learning systems learning with (a): Grid search combinations of first and second layer nano-synapse number of addressable levels/bits for systems learning with linear/quasi-linear devices (b): Grid search combinations of first and second layer nano-synapse number of addressable levels/bits for systems learning with strongly non-linear devices.

Practically, this result implies that, when building multi-layer memristive ANNs, priority ought to be given to constructing earlier layers with higher quality nanosynapses (nanodevices), wherever possible. Theoretically, this result relates to an emerging information-theoretic understanding of the way that deep networks learn known as the information bottleneck principle. More precisely, multi-layer neural networks resemble a Markov chain, and information lost in a preceding (earlier) layer in the network cannot be recovered by a later layer in the network [270]. This implies, in the case of our networks, that more information is lost in the forward passes (inference) (which pass through the g_1 layer first) than in the backward or back-propagation passes (which pass through the g_2 layer first). However, the analysis we have presented so far is necessarily incomplete, since in addition to nano-synapse depth there are two critical information representations being held, the vector of the hidden layer neuron activations and the vector of the output layer neuron activations. In the future, it would be interesting to see what variety of on-chip neuron designs can alleviate this critical constraint.

3.4.3 Resilience to non-adaptive devices

Occasionally, individual devices are unadaptive, rendering them conventional resistors; this phenomenon is often referred to as stuck-on/off defects. To assess the effect of this phenomenon, a set percentage of all, in the case of single crossbar or gradient systems, or second layer in the

case of NoProp, devices remain constantly at either G_{ON} or G_{OFF} (are unadaptive) throughout the entire training and testing process. These simulations were repeated ten times with different broken systems and random initial conductances and averaged, with standard deviation visible. As visible in Figure 3.16, the introduction of broken devices only slightly harms systems built with linear devices, but can substantially harm systems built from non-linear ones, across all architecture choices. Second, multi-layer (NoProp, backprop) systems are substantially more resistant to this effect than the one-layer or crossbar systems. While multi-layer/multiple-crossbar systems harness a non-linear projection and the interdependence of the first and second layer to reduce reliance on any one individual device, single crossbar learning systems at best try to constitute a two-dimensional reconstruction (image) of the class. In this context, the removal of individually adaptive devices, while at first insignificant, becomes increasingly catastrophic in single layer systems beyond a certain point.

3.4.4 Resilience to programming mode asymmetries

Next, an asymmetry in the device's programming modes (SET, RESET) was simulated; this could be either an intrinsic issue of the device, as often encountered with devices that have asymmetrical physical mechanisms between creation and destruction of filaments ([271], as seen with the TBFe devices in Section 2.2, or a system failing due to unreliable programming circuitry or sneak paths. Following the device case, we modeled increasingly asymmetric (more powerful) RESET as compared to set, where all programming pulses sent are equivalent but the resulting effect on negative conductance evolution is constantly stronger than positive evolution ($\Delta G_{RESET} = \zeta \Delta G_{SET}$). The effects as ζ increases are visible in Fig. 3.17 for systems with rich nanosynapses ($g = 256$). All systems are notably more harmed by the conductance mode asymmetry effect than any other effect, with one-layer systems being the least resilient, backprop the most resilient, and NoProp an intermediate case. The fragility of one-layer systems relates to the combination of smallest system and simplest programming style, while backprop systems benefit from the most complex programming style.

3.4.5 Energy scaling tradeoffs

Assuming an analog array of nanosynapses consisting of N columns and M rows, the overall energy footprint for both reading and writing is given as:

- Parallel read and/or inference steps should scale as NxM (i.e, the ability to perform vector-matrix operations) multiplied by the elementary forward/read cost E_{read}.

- Parallel write should scale as NxM for both simultaneous and sequential cases multiplied by the elementary writing cost E_{write}. When writing in the column-by-column fashion, we assume that diodes or another variety of high-quality access device has been paired to each output neuron so as to minimize parasitic current losses [208, 272].

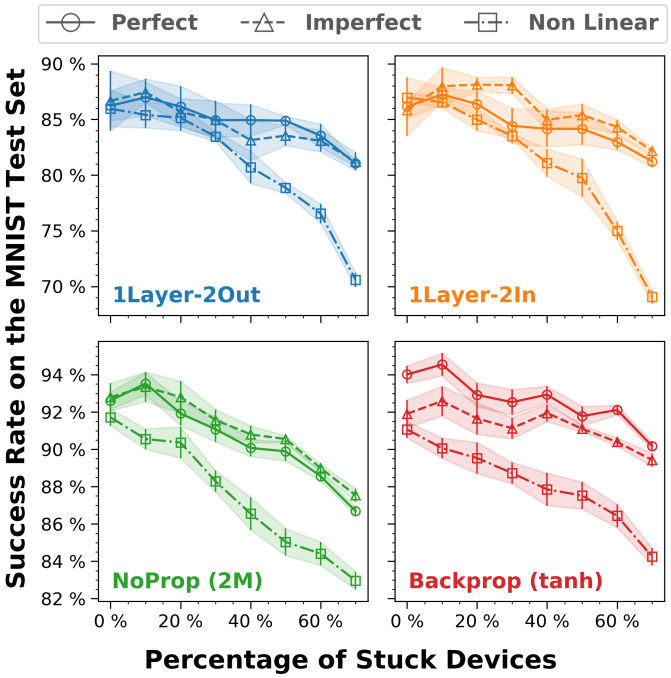


Figure 3.16: Effect of unadaptive synapses on learning outcomes, where given percentage of devices are stuck at either maximal or minimal conductance.All systems have been given sufficient epochs to converge; $M = 3000$ for NP, $M = 300$ for backprop. Every point is the mean of $s = 10$ sample systems, with the shaded standard deviation shown.

We computed read/inference and programming costs for three different device test cases: our TBFe devices described further in Section 2.2, the ENODe polymer devices given in [273], and the ferroelectric devices given [274]. The following computations are based on the following energy dissipations estimates

$$E_{\text{read}} = T_{\text{read}} G_{\mu} (V_{\text{read}})^2 \tag{3.14}$$

$$E_{\text{write}} = T_{\text{write}} G_{\mu} (V_{\text{edp}})^2 \tag{3.15}$$

Where V_{edp} is the voltage difference seen across the device as the result of the primary and associate programming pulses, and G_{μ} is an average conductance for the considered device species. Notably, the values for the writing programming pulses for TBFe, ENODE, and ferroelectric devices are $81nJ$, $2.92nJ$, and $1.481pJ$, respectively. Based on these values, we have extrapolated elementary operation costs to budgets for complete learning (convergence).

Among all systems, the primary energy cost (> 90%) is due to write or programming operations to adapt the nanosynapses. In between all compared systems, one-crossbar environments expend the least energy to converge and gradient learning systems the most, with No-Prop learning systems as an intermediate case (Fig. 3.18 (a)). Among multi-layer learning systems, a notable trade-off exists between the energy-savings of a quick finish (faster conver-

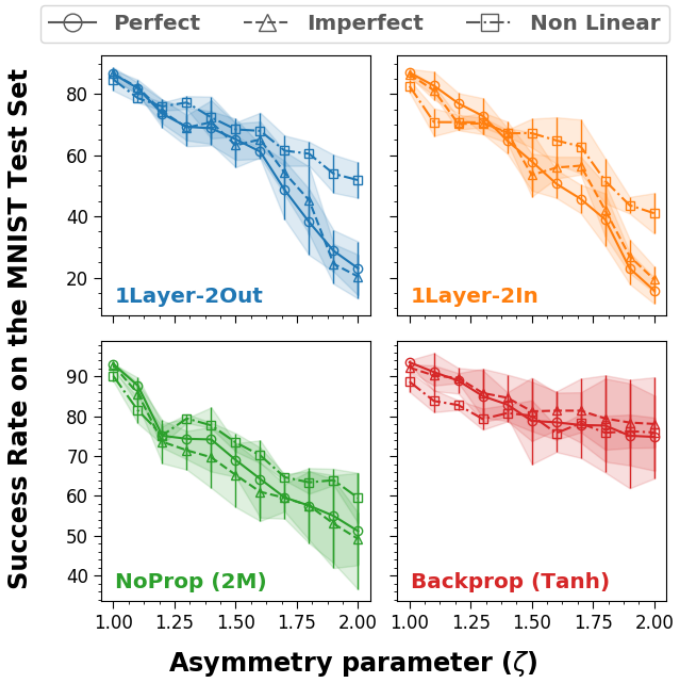


Figure 3.17: Effect of disturbed programming on learning outcomes, where systems have RESET operations grow asymmetrically large at the given constant level. All adapting synapses have equivalent richness ($g = 256$). All systems have been given sufficient epochs to converge; $M = 3000$ for NP, $M = 300$ for backprop. Every point is the mean of $s = 10$ sample systems, with the shaded standard deviation shown.

gence), and the energy loss from system overhead/size (No Prop systems depend on a larger M). This trade-off was explored further and our analysis suggests that, up to $M = 6300$, NoProp systems gain more from the quick finish than what they lose from the larger dimension (Fig. 3.18 (b)). While these trends are independent of the chosen device's elementary programming costs, the choice of nanosynapse can have significant implications on the final energy costs of learning. All extrapolated results are visible in Table 3.3 both for standard stochastic gradient descent, as well as batch-style programming. Batch style programming results in very substantial energy savings, since it directly reduces the number of programming pulses needed to implement learning. However, as additional analog components are needed to integrate and store intermediate values between each program step and these have power budgets as well, these gains would be somewhat mitigated in hardware implementations.

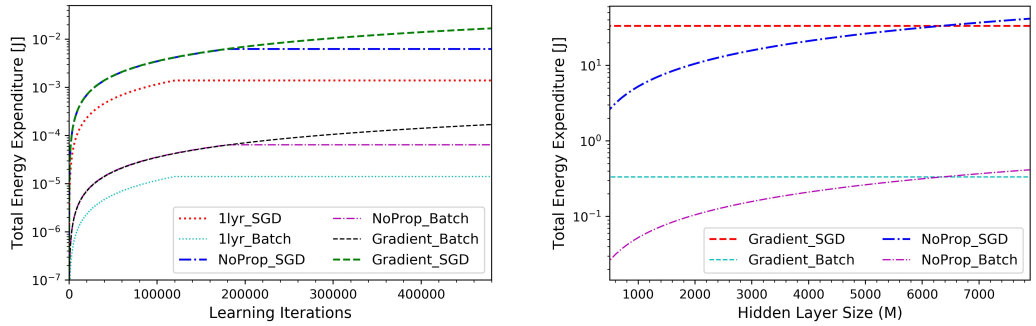


Figure 3.18: Energy expenditure for all considered architectures is estimated as a parameter of system speed (a) as well as size (b). Energy estimates are computed by multiplying the number of V_{prog} and V_{read} pulses required according to the architecture/ learning policy, times the total number of iterations. Batch-mode estimations calculate read energy at every iteration, but V_{prog} pulses applied to the system only once every b times; $b = 100$ throughout. Once converged, no additional read or programming pulses are applied; one-layer systems converge in 2 epochs, NoProp in 3, and gradient in 8.
(a) depicts increasing energy use for all considered policy and assumes the device from [274]; (b) varies in M for No-Prop systems, using gradient as benchmark, and assumes device from [273].

Nanodevice ANN Algorithm	**Total Energy Cost of Learning [Joules]**		
	Polymeric - TBFE	**Polymeric -ENODE**	**Ferroelectric**
One Layer SGD	76.2	2.74	$1.4e-3$
One Layer Batch	0.76	0.027	$1.4e-5$
NoProp SGD (M=3L)	342.96	12.36	$6.3e-3$
NoProp Batch (M=3L)	3.466	0.124	$6.3e-5$
Backprop SGD (Tanh, M=300)	920.7	33.2	0.017
Backprop Batch (Tanh, M=300)	9.2	0.33	$1.7e-4$

Table 3.3: Energy cost for three different nanosynapses integrated into our learning nanosystems are estimated: fast, high-voltage TBFe polymeric device (Section 2.2); slow, very low-voltage ENODE polymer devices ([273]); and very fast, intermediate voltage ferroelectric memristor devices ([274]. All energy computations are based on a characteristic convergence times of 2 epochs (single layer), 3 epochs (NoProp systems), and 8 epochs (gradient learning). NoProp Systems are M=3L; backprop systems M=300; all backprop systems use $tanh$ activation. For all cases, the batch learning size of $b = 100$ is considered (total programming operations are reduced one-hundredfold).

3.5 Conclusion and Perspective

Our benchmarking analysis uncovered intriguing trade-offs that could be of great interest to neuromorphic engineers building future systems with nanosynapse arrays. In the following three sub-sections, we summarize the contrasting advantages of the two systems relative to each other before finally highlighting which constraints or limitations confronting each individually.

3.5.1 Contrasting Properties

Speed

1 layer and NoProp systems learn quickly, often reaching 90% of their final recognition rate in a single epoch, and full convergence between 2-3 epochs. In contrast, sign-based gradient learning systems are slow, requiring between 6-9 epochs to completely converge.

Size

A major dis-advantage of NoProp hardware neural networks is the large physically realized hidden layer. For instance, the best-performing system here uses a hidden layer with approximately ten times as many hidden layer components as a gradient learning system that performs at least as well.

Required nanodevice quality

One of our central conclusions is that the stringency for nanodevice quality in gradient learning systems is far higher than in the NoProp/ELM systems. Nanodevice resolution of 6-7 bits is required even with linear device, and for non-linear ones, 7-8 bits of weight space is required. On the other hand, ELM system can respectfully perform with as little as 4 bits (16 states) and perform with full accuracy with 5-6 bits (32-64 states). This result is broadly consistent with the literature; in [275], 4 bits were shown sufficient to implement weights executing simple STDP (temporal correlation) learning; in [276], a memristive two-layer system using an unsupervised first layer operating according to the locally competitive algorithm, and a supervised second layer again required between 4-6 bits depending on the ideality of the device.

Resilience to non-linearity

Relative to all other systems, NoProp loses the least when non-linear device behavior is accounted for; backprop and simple regression systems lose two and three times as much performance, respectively.

Accuracy

On the other hand, gradient learning systems with the most complex hidden layer transfer system $tanh$ undeniably outperformed all other learning systems, achieving 97% on the MNIST task. However, this level of performance was only reachable with the more complex /high-overhead mb-SGD programming style.

Energy footprint

Energy costs are again driven primarily by the programming overhead, and demonstrate a major tradeoff between system size and required system training length. Of all systems the one crossbar learning systems cost the least, and between multi-layer systems the NoProp systems gain more from the fast finish than from their large size. Across all the architecture choices, batch learning can greatly reduce the total number of programming steps, but it comes with non-negligible overhead.

3.5.2 Constraints on effective local gradient learning

Our on-chip gradient learning systems learn equivalently to the software comparison system when eight bit weight resolution is employed, and do progressively worse as they are further discretized. This is consistent with the machine learning literature, which has suggested that less than 8 bits can significantly impede network generalization and accuracy [156]. Recently, a contrasting binarized implementation for deep neural networks was proposed [268], which binarizes neuron activations, and weights. However, this approach requires the accumulation of a highly analog gradient value at the end of the batch to faithfully implement stochastic gradient descent. Future highly dense implementations of gradient learning, therefore, will have to store a highly analog value somewhere to learn properly; we have proposed doing so intrinsically (within the device itself).

Other works using gradient learning in the crossbar environment have achieved slightly superior results than ours; in [277], 98.5 % was achieved on the MNIST test set. This is around 1.5% better than our best performing system (batch mode $tanh$). In explaining the higher performance, we found that this work employed an analog nanosynapse model with extremely high nano-device weight resolution ($g > 1000$). Another study on hardware acceleration of deep neural networks reached 97% using a stochastic simplification of the backprop learning rule [278], which is equivalent to our results. Like in our work, the authors considered g as a parameter, and found that when devices possessed less than $g = 600$, results fell substantially. These comparisons highlight the importance of including realistic nanodevice richness constraints when estimating the possible performances of future learning systems.

3.5.3 Constraints on effective local 'NoProp' learning

The principle dis-advantage of the NoProp policy relative to the other two choices is its high system size/overhead, as hidden layers multiples of input dimension are necessary for useful projection. To mitigate this requirement, designers might:

- Compress the dimensionality of input data in a pre-processing step, using PCA or t-SNE. A pre-processing network, such as an on-chip restricted Boltzmann machine [279], or an addition crossbar performing data clustering [280], could perform this locally.

- Encode the input as channels in the time domain, although additional circuitry is re-

quired at the hidden layer. This approach is discussed further in Chapter 4, Section 4.2.

In all cases, three-dimensional crosspoint array operations could further lessen the size/scope of hidden layer operations. For instance, 2M biasing could be obtained via current differences across two random layers, rather than a single crossbar array with double the dimension

Finally, maximum accuracy obtained in any of our two-layer projection systems lags behind competing gradient learning system alternatives. However, in [127], various improvements to standard systems, such as computed inputed weights (CIW), receptive fields, and extending to several non-linear projection layers to 2 (3 total hidden layers), were demonstrated to collectively boost ELM performance to achieve only 1% loss on the MNIST test-set. This suggests that in addition to 'standard ELM', mapping the 'enhanced ELM' concepts discussed in greater depth in 1.3.7 to a nano-electronic environment could be a fruitful approach. In the following Chapter, we consider two related concepts.

3.5.4 Considering timing in multi-layer context

The feed-forward multilayer architectures considered so far have conducted learning in a very ordered and discretized way, such that plasticity effects on the device level, and systems-level timing parameters, could be carefully controlled or ignored, respectively. Although this can indeed improve the predictability of learning operations, it may also pose an additional bottleneck in performance as it restricts these systems from harnessing intrinsic temporal dynamics. These dynamics allow recurrent and reservoir computers to generalize well on tasks with which multi-layer feedforward networks greatly struggle (see Secs. 1.3.8, 1.3.9). In the following chapter, we push multi-layer memristive systems to incorporate timing parameters, at either the device and the systems level.

Chapter 4

Temporal Memristive Learning Architectures

Space and time are the framework within which the mind is constrained to construct its experience of reality

Immanuel KANT

"THIS CHAPTER *demonstrates larger on-chip architectures that, at some point in their operation, use the intrinsic time-dynamics of a circuit or device as a key component of learning. This unlocks new frontiers in performance, and provides important stepping stones towards the ultimate vision of building nanosynaptic arrays that learn on-chip with the full computation power of full Recurrent Neural Networks.* "

THIS CHAPTER focuses upon the integration of memristive devices into learning systems which use timing parameters at various levels to perform local learning. In contrast to standard feed-forward models, bio-inspired learning models that integrate time as a key part of the learning process may lead to new frontiers in accuracy as well as energy efficiencies. In particular, the earlier described NoProp-inspired nanodevice learning systems are enhanced in two critical ways. The outline for this final chapter is as follows:

1. First, a general motivation for exploring timing dynamics as a feature of future neuromorphic computing systems is provided, with a focus on inspiration from the field of neuroscience and how it relates to improvements in conceptual neural network models. (4.1).

2. Second, we present and evaluate the performance of a multi-layer learning scheme which combines a time-intrinsic computing portion and a standard supervised learning portion. This "enhanced ELM" system outperforms the standard random weights multilayer system explained previously in 3.2.2. Off-chip and on-chip learning schemes are both considered. (4.2).

3. Third, we present and evaluate a reservoir computing inspired scheme appropriate for on-chip implementation with analog nanodevices. By adding time-specific complexity at the hidden layer, performances greater than any of our previously demonstrated multilayer ANN systems are realized, and with less overhead too. (4.3).

4. Lastly, a discussion of the outcomes and prospects for further improvements is made.

4.1 Motivation for exploring temporal learning and metaplasticity

4.1.1 Spatio-temporal learning in the brain

The brain is, without a doubt, a spatio-temporal learning machine. That is, the timing of consecutive learning operations and the intrinsic dynamics of neural circuits are temporal in nature. In particular, temporal integration at varying time-scales seems to be critical at both the intra-neuronal level (e.g., learning in dendritic branches) [94, 95], as well as the inter-neuronal level (e.g, firing patterns and spatio-temporal filters in populations of neurons) [281, 282]. In order to build these sort of features into learning systems, a general mathematical model is needed. Some foundational model for spatio-temporal analysis in neural circuits include the linear-non-linear model (LN) [283], which can be generalized into many pathways, as well as the generalized linear models (GLM) [284]. Recent improvements to these models use regularization and hierarchy to more accurately predict the spiking behavior of multi-layered neural cirucits in the retina [285]. In all of these cases, the power of the model relies upon staged

computation between time-dependent spatio-temporal filters or kernels, followed by time-independent static non-linear filters.

Drawing upon these models of computation and attempting their integration into artificial neuro-synaptic learning systems can provide a significant source of inspiration to existing computational methods, while also inversely informing a deeper understanding of neuro-computation [286]. Focusing on the synapse as a laboratory for implementing some of these ideas is a powerful tactic to achieve this strategy, given the richness of behavior and dynamics even at this small scale [287].

4.1.2 Spatio-temporal effects: a natural extension of NoProp systems

As mentioned in Sec. 3.1, NoProp systems arguably possess more neuro-inspiration than competing machine learning approaches [1]. In general, NoProp systems have a surprising ability to achieve quick convergence on high-dimensional problems, in spite of the apparent simplicity of their learning style. Unlike the brain, however, standard NoProp systems still fail to either account for time-dependent plasticity effects, or to implement crucial temporal or spatial coding tricks implemented by the brain. Nevertheless, enhanced ELM/NoProp systems are a natural environment in which to implement these methods (as also discussed in Sec. 1.3.7.1). In the remainder of this Chapter, two such improvements are made: first, a critical plasticity improvement to the NoProp learning system, in particular its first-layer weights, is proposed and evaluated in Sec. 4.2; second, a major improvement to its hidden layer functionality and complexity is proposed and evaluated in Sec. 4.2. However, before describing these schemes, we note further motivation and inspiration behind the two considered extensions.

4.1.2.1 Inspiration for expanding NoProp with plasticity transitions

In the brain, short-term plasticity (STP) refers to synaptic state change (potentiation) connecting neurons on the scale of seconds to minutes, while long-term plasticity (LTP) potentiates synapses for hours, days or even for a lifetime. Meta-plasticity learning systems or mechanisms [288] use learning taking place over different time scales in order to implement critical functions such as memory consolidation. Often times, complex metaplastic effects take place over physically separated memory systems, often which learn at different rates or in different ways [289]. In particular, priming is a cognitive mechanism in which an associate cortical system or pathway is trained to pre-recognize or prepare a primary memory system to perform a later task or association [290].

While the transition from short to long-term plasticity is already considered in neuroscience models, this transition remains an under-explored topic in the field of memristive learning due to an interest in the non-volatility of devices. This is peculiar since, just as synapses in the

[1] As far as we presently understand, biological neuronal networks are not implementing the backpropagation algorithm; on the other hand, some neural coding strategies look vaguely reminiscent of NoProp approaches

brain, some memristive nanodevices present rich plasticity behaviors over shorter term time scales [164, 291–293]. While [291, 292] explored memristive STP/LTP transitions and [294, 295] realized metaplasticity effects with memristive devices, none of these works developed learning architectures based on these mechanisms. In [296], volatile tungsten-based memristive devices that relax relatively quickly were considered for integration into a learning system, but only the STP regime exploited for classification. Given the gap between the potentials of nanodevices to intrinsically implement plasticity transitions, and the well-established importance of these transitions to efficient learning (memory) systems, we are excited to explore how such a transition can improve a nanodevice NoProp system.

Concretely, we do this by priming the first layer weights. In deciding to implement the plasticity transition in the first layer, we additionally drew upon inspiration in the machine learning literature, which tended to emphasize the effectiveness of priming or pre-computing the first layer weights of ELM/NoProp software systems. Indeed, this computed input weights (CIW) strategy was reported to improve performance over the standard case [127, 297]. Here, we show a similar result subject to unique device timing constraints.

4.1.2.2 Inspiration for expanding NoProp with hidden layer complexity

With regards to the latter strategy, the Neural Engineering Framework (NEF) proposed by Chris Eliasmith [100] looks in a sense like a critical enhancement of standard NoProp systems. In addition to a random weights encoder (first layer) and a trained decoder (second layer), the hidden layer uses different neuronal tuning curves, or the fact that neurons have different preferred stimuli. This principles is used to achieve very general results. Recently this framework has also attracted perspective from the neuromorphic engineering community [298]. In addition to tuning curves, hidden layer effects can also be enhanced by considering various time-dependent computing kernels at each neuron [128].

4.2 On-Chip Plasticity Transition Learning Schemes

Using the transition from short-term to long-term plasticity as the core component of a nanoelectronic learning system is the crux of this first work. Two different learning architectures that use this transition to memorize and retain target data are detailed, and simulated versions of these systems are then used to attempt two classification tasks. In particular, the two models are a single crossbar approach, and multiple crossbar approach partly inspired by the NoProp system. The latter system in particular exhibits exciting performance and high resilience to device variability. All of the following results are simulated.

4.2.1 Nanosynaptic Plasticity Transitions: Device and Model

Here, we focus on the integration of a highly promising device, electrochemical metallization (ECM) cells where rich STP and LTP dynamics have been evidenced recently [293]. We were introduced to this device and its exciting properties through a scientific collaboration with Drs. Fabien Alibart and Selina LaBarbera at IEMN Lille, who have experimentally investigated the intrinsic time dynamics of custom ECM nanodevices. In this collaboration, we drew upon an experimentally validated model of their behavior to consider and construct larger nanoelectronic systems.

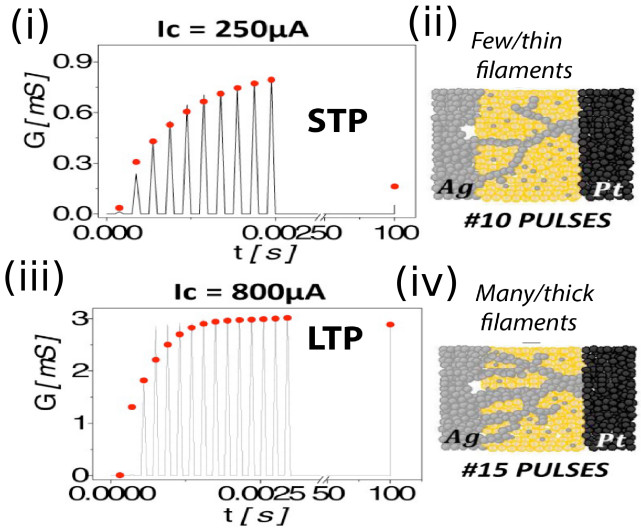


Figure 4.1: (i) depicts a pre-synaptic spike train that keeps a device in the STP regime; a corresponding, weaker filament that can easily relax is pictured (ii). (iii) depicts a more powerful spike train that successfully moves the pictured device from the STP to LTP regime with many thick filaments (iv). The black solid lines in (i), (iii) are the measurement results and the red dots are the model predictions.

The devices considered are electrochemical metallization (ECM) cells with a 60 nm switching layer, where dendritic filaments form in between a reactive top electrode (anode) of silver, and an inert bottom electrode (cathode) of platinum [293]. The application of a positive bias above a threshold V_{th} causes oxidation and drift of silver ions (Ag^+) across the Ag_2S switching layer from the cathode towards the anode. This increases conductivity and physically corresponds to the formation and strengthening of filaments. Conversely, a negative bias induces reduction at the Ag electrode, weakening filaments and decreasing conductivity. This bipolar behavior is different than the unipolar behavivor of the TBFe devices introduced in Chapter 2.

Moreover, unlike many nanosynapses which retain their states for long periods (are intrinsically non-volatile), the ECM cells shows a natural relaxation towards lower conductivity as Ag^+ ions continue to diffuse and reverse oxidation-reduction occurs. Critically, this natural relaxation *may be fast or slow*, depending on the quantity and quality of the filaments. As reported in [293], varying filamentary morphology and a possible trade-off between filament density and

diameter create complex synaptic behavior. In particular, the transition from a relatively small relaxation time (τ)- the STP regime- to a larger τ corresponding to the LTP regime - was tunable both by the number and the characteristics of subsequent pre-synaptic excitatory pulses. Fig. 4.1(i) depicts the STP case where a small number of pulses strengthen the filament so that the ECM cell's conductance increases to $0.9mS$. However, this state is not stable: after a time $\tau = 100s$, the conductance has relaxed to a low conductance state. Conversely, Fig. 4.1(ii) depicts the LTP case. Differently timed spikes move the synapse to a high conductance of $3mS$, which remains stable after $\tau = 100s$.

A detailed model of STP to LTP transition in ECM cells, reminiscent of a biological model of plasticity [287], was validated experimentally in the previous work of our collaborators [293]. We now revisit the basic equations of this model. Synaptic potentiation increases in response to a train of pre-synaptic pulses; the facilitating time constant τ_{fac} constantly increases as spikes are applied and conductance increases, facilitating the STP to LTP transition. After each programming spike:

$$\tau_{\text{fac}} = a \cdot G(t)^b, \tag{4.1}$$

and after any given delay Δt from the last spike at time t, the conductance is

$$G_{\text{relax}} = G(t) \cdot \exp\left(\frac{-\Delta t}{\tau_{\text{fac}}}\right) \tag{4.2}$$

Finally, at time $t + \Delta t$, the conductance value results from the sum of the exponential relaxation and of a programming spike if any is applied:

$$G(t+\Delta t) = \begin{cases} G_{\text{relax}} + U(A - G_{\text{relax}}) & \text{, if spike} \\ G_{\text{relax}} & \text{, if no spike.} \end{cases} \tag{4.3}$$

A corresponds to maximum synaptic efficiency (G_{max}). A typical value extracted from device measurement is $A = 4\text{mS}$. The typical synaptic efficiency is $U = 0.025$. The power law prefactor $a = 2.42 \times 10^{-12}\,\text{s} \cdot \text{S}^{-\text{b}}$, and the power law exponent $b = 4$ [293].

4.2.2 A Simple Learning Task and Algorithm

In the following section, we detail an architecture that highlights the promise and challenge associated with exploiting the STP to LTP transition in nanodevices. At this stage, the architecture has three components: a software image database or sensor and circuitry to convert pixels into voltage spikes, an all-to-all crossbar that connects input and output neurons electrically at each ECM cell crosspoint, and accompanying CMOS circuitry. Learning occurs in three stages.

4.2.2.1 Imprinting

In the first stage, ECM cells are "imprinted". This stage is sub-divided into several epochs corresponding to the total number of classes J.

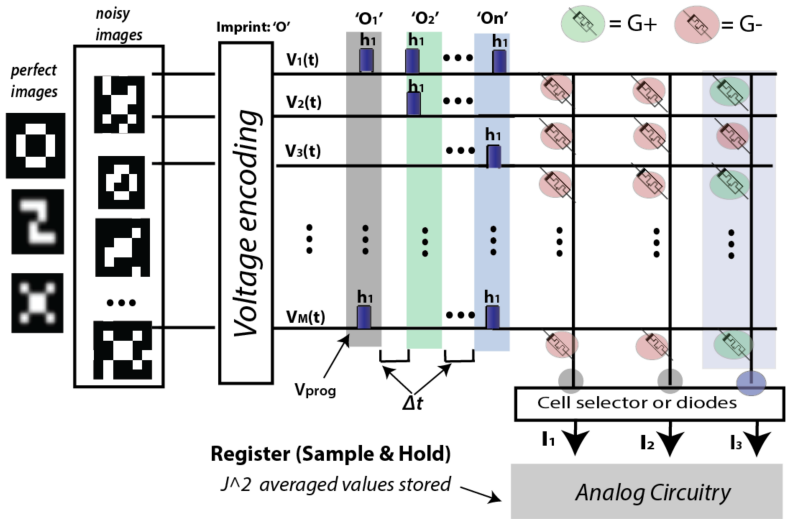


Figure 4.2: The simple learning system, with characteristic images input to the system, selected and non-selected nanodevices, and input and output computing accessories required for learning.

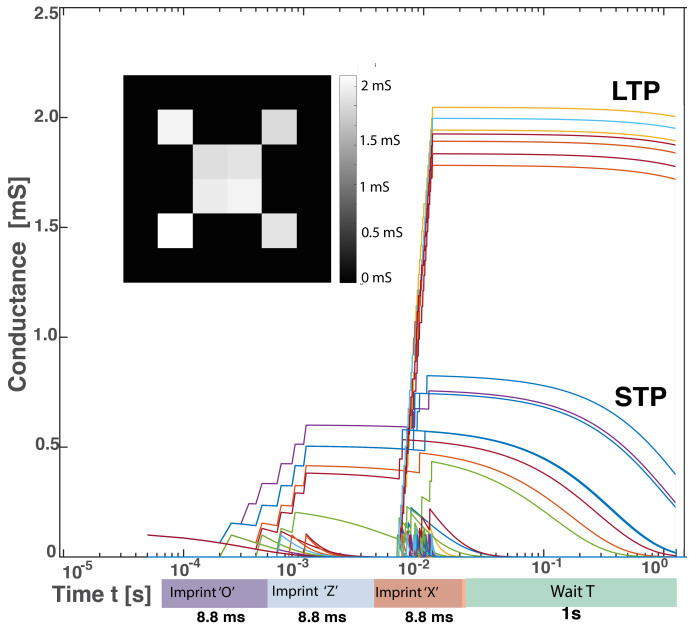


Figure 4.3: Conductance as a function of time during the imprinting process- $n = 30, \Delta t = 200\mu s$ and subsequent waiting period, for the 36 ECM cells connected to a single output neuron. Inset: conductance of the cells at t = 1.026 s, presented as a reconstructed 2-D image. Note time is portrayed on a semi-logarithmic and not linear axis.

During each epoch, n noisy examples of the given class are subsequently presented to the crossbar where an active (white) pixel is presented at $V_{\text{in}} = h1$, and inactive (black) pixels are silent. Each input neuron receives one pixel consistently, and the patterns are presented with a delay Δt as depicted in Fig. 4.2, where (A) is in epoch 'O' and (B) in epoch 'X'. After imprinting, no voltage is applied on the crossbar for a wait period T. This allows ECM cells in the STP regime, assumed to be noise, to return to low conductance states, while it will not affect those in LTP. Fig. 4.3 presents the evolution of the conductance of the 36 ECM cells connected to one particular neuron, during imprinting ($t_{\text{imprint}} = 26.4\text{ms}$) and subsequent wait ($T = 1s$). The final conductance map for the output neuron- in this case it has learned 'X'- is visualized pixel-by-pixel in the inset.

Imprinting only works if each output neuron corresponds to a different class. In the present work, output neurons are mapped to classes in a supervised manner- employing a selector as discussed in Section 1.4.4.1. Only nanodevices at the intersection of an active pixel/row (receiving pre-synaptic spike $V_{\text{prog,h}} = 0.42V$, $V_{\text{prog,w}} = 100\mu\text{s}$) and the selected column (green nanodevices in Fig. 4.2) increase conductance.

4.2.2.2 Collecting and testing

Second, images are presented to the network in 'read' mode ($V_{\text{read}} = 0.1V$) - so as to not disturb conductances- and currents are read out at all output neurons (not just the corresponding one). As output currents are a dot product of device conductances and active pixels for a given image, many unique values are possible. These values are stored in a circuit (register) below where they are iteratively averaged. After N examples, $J^2 = 9$ currents (signatures) are stored: I_{reg}. Computing an iterative average during training and storing it in a register may be achieved in either analog (operation amplifiers, sample and hold circuits each containing a capacitor), or digital (analog to digital converters and conventional digital memory, eg RAM) fashions, or a combination. Agnostic of implementation, the circuit overhead is significant and a considerable drawback of this approach.

Testing is the final phase: K unknown digits are presented at V_{read} and output currents I_{test} are compared to the register's values. The predicted class is the one which minimizes E_{tot}:

$$E_{\text{tot}} = \sum_{i=1}^{J} \left| I_{reg_i} - I_{test_i} \right| \tag{4.4}$$

If predicted class is the true class, '1' is stored in an accompanying memory cell; else '0'. The final score is the sum of all correct guesses g divided by K. Computing E_{tot} during tests additionally requires an absolute value circuit. Each output neuron (class) must have access to equivalent circuitry.

All of the following reported results, as well as that depicted in Fig. 4.3, were produced by a software program that simulates a crossbar of nanodevices, each following the mathematical model for conductance evolution introduced earlier, and tracks evolution of synapses and cur-

rents over time in response to voltage encoded input spike trains. This simulation software also models nanodevice specific issues such as device variability. For the simulation results immediately following, $N = K = 100$, long wait $T = 1s$. Noise is added by randomly flipping to their opposite state 10 % of all pixels in images used for imprinting, training, and testing.

4.2.3 Calibration on the Simple Task

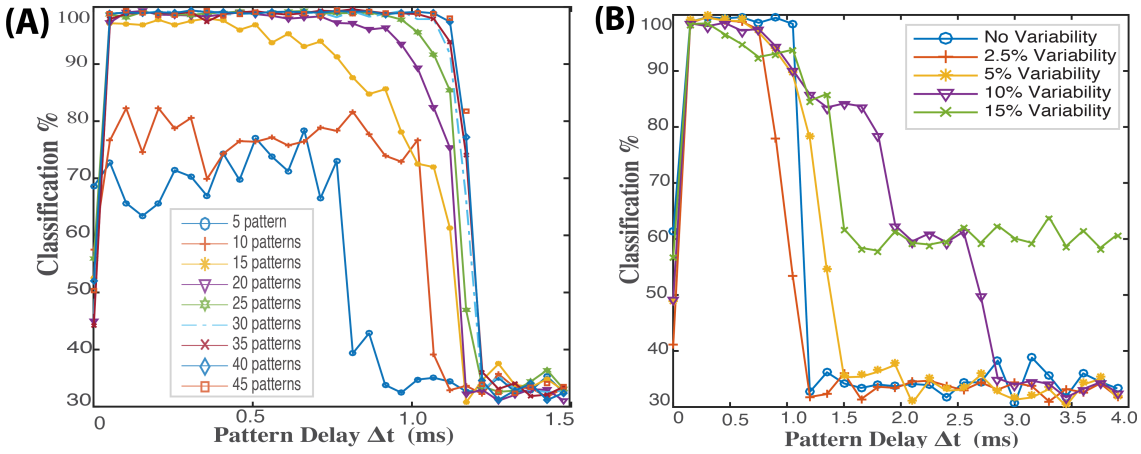


Figure 4.4: Both plots demonstrate classification rate of the simple system on the simple images, as a function of time step Δt. (A) for different number of patterns presented per epoch (class) and (B) for different degree of device variability ($n = 45$ patterns imprinted per epoch).

To calibrate the ECM learning system, we first used a simple, custom-constructed image dataset in Matlab. It consists of $J = 3$ classes, each a 'hand-written' letter: 'O', 'Z' and 'X'. Each image contains $L = 36$ total and 8 active pixels. We automatically generate a training set from the perfect images by randomly adding noise. To add noise, we apply a probability at every pixel that it may switch state; that is active pixels (1) may switch to inactive (0) ones, and the opposite case.

Fig. 4.4(A) presents classification rates as a function of the chosen inter-pattern wait step Δt. Each series represents a different number of patterns n presented per epoch of imprinting stage. The performance reaches a nearly perfect 98% over a broad timing range (0.1ms < Δt < 1.2ms) for $n > 20$. Below $n = 20$, it is not possible to reach the plateau as there are insufficient presynaptic pulses to move nanodevices from the STP to LTP regime (also visible in Fig. 4.1(i),(ii) and discussed in [293]).

Fig. 4.4(A) also shows sub-optimal classification before and after the optimal range. The former occurs when patterns imprint too fast to synchronize with the nanodevice's normal relaxation parameter τ_{fac}, hence the conductance map is over-saturated (too many devices enter LTP). Conversely, when Δt is too large, insufficient devices enter LTP to retain the digit image. In these cases currents no longer vary meaningfully neuron-by-neuron, and classification becomes difficult.

4.2.3.1 Effect of Device Variability

Next, we consider the case where ECM cells each behave slightly differently to equivalent presynaptic spikes. Each now receives a different internal device timing variable: U, A, a. From Eq. 4.1, each device then possesses a slightly different τ_{fac} as a changes; from Eq.4.3, each conductance evolves a bit differently due to varying synaptic efficiency (U) and $G_{\max}$ (A). Random values U, A, a are drawn from a normal distribution with mean (μ) set as those listed in Section II, and coefficient of variation considered over degrees $\sigma/\mu = \{0.025, 0.05, 0.1, 0.15\}$. For each degree of variation, 20 simulations were performed at each Δt value and averaged. As depicted in Fig. 4.4(B), increasing variability reduces the nearly perfect classification plateau. Unlike the uniform case, increasingly variable crossbars do not fall off a performance 'cliff', but experience a gentler landing at increasing dispersion parameters. In the $\sigma/\mu = 0.15$ case, the same recognition 'floor' as in the other cases (30%) is not reached even at a very long inter-pattern wait ($4ms$).

4.2.3.2 Effect of Increasing Training Samples

Few examples are needed in order to learn the functions, because the register reaches a useful average very quickly. In the uniform case classification reaches 70% with 5 samples, and approaches 100% after 10. Considering variability, 15 samples are needed to reach peak (95%) classification; however, the highest dispersion case (15%) takes 25 samples to reach its peak (90% classification).

4.2.3.3 Effect of Increasing Noise

In the uniform case, classification remains nearly perfect until around 15% of pixel flips and then deteriorates linearly after that until a minimum of 75%. Low and medium variability cases perform well until 10%, following a similar deterioration trajectory thereafter. However, the highest variability case (15%) only does as well as the others in the 0-5 % noise range; by 20% noise-induced flips it already falls to 80% correct.

4.2.4 Calibration on the MNIST Task

In attempting to classify the MNIST database of hand-written digits [151], $J = 10, M = 784$ so 7840 synapses (ECM devices) attempt to resolve the problem. In this case, the register must hold $J^2 = 100$ values. While MNIST provides $N = 60k$ training, $K = 10k$ tests, only $N = K = 1k$ were used. Overall, the system's performance on this task is not favorable. Fig. 4.5(A) shows that classification peaks at $\Delta t = 1.1ms$ with 61% correct. The insets highlight that at this peak, digits are relatively well constructed if sparse, while reconstructed pixel maps in the mostly evaporated (super-optimal Δt) and oversaturated (sub-optimal Δt) regions are unusable. However, even at the peak, classification is weak because of an intrinsic algorithmic weakness. Since

Variable	Definition	Default Value
L	input pixels	36 (simple) 784 (MNIST)
M	hidden layer size	n/a
J	output neurons	3 (simple) 10 (MNIST)
N	training patterns	60,000 (MNIST)
K	testing patterns	10,000 (MNIST)
n	patterns per imprint cycle	50
Δt	Inter-pattern wait	200 μs
T	Rest/wait period	1 s

Table 4.1: Summary of key parameters and variables used in system calibration

all the register memorizes are currents and test images have a wide variety of active pixels (in contrast to the small images), it has a hard time distinguishing between different classes that produce similar current sums. Fig. 4.5(A) also suggests that increasing the number of output neurons beyond $J = 10$ does not help with the present algorithm, as stored averages for redundant neurons will be similar. Fig. 4.5(B) shows that device variability has a deleterious effect on learning ability. At low dispersion, degree peak classification drops to 50% while preserving a similar trend, while large dispersion echoes the phenomena of resilience to large pattern delays observed in Fig. 4.4(B).

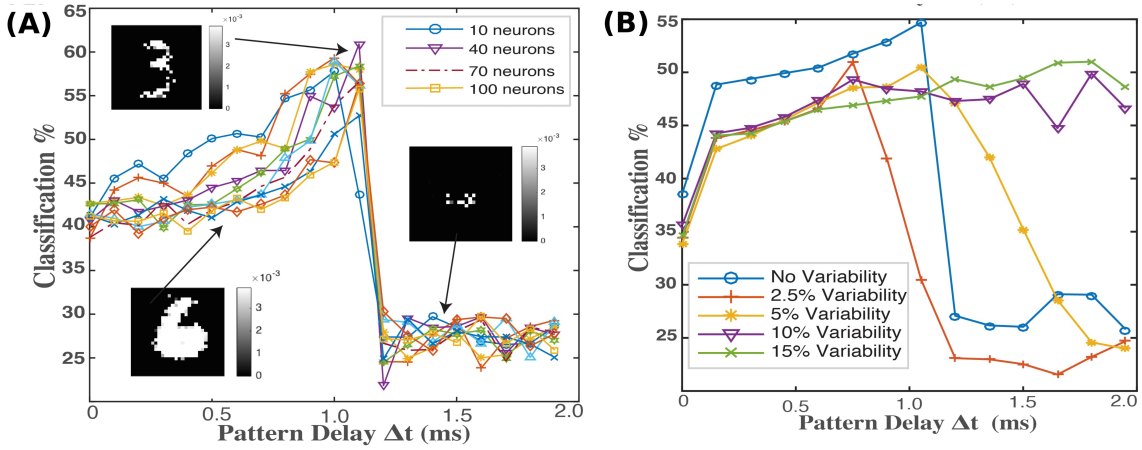


Figure 4.5: Both plots show classification rate on the entire MNIST test-set also as a function of Δt. (A) Varying J output neurons when constantly $n = 50$. The three insets depict reconstructed conductance maps for one of the (J_i) neurons imprinted at given Δt parameter; white represents device in LTP (ON), and black those in STP (OFF) (B) Effect of degree of device variability ($J = 10$ output neurons,$n = 50$ in each case). Every image has 10% noise.

4.2.5 Multi-Layer Plasticity Transition Learning system

Inferior performance on the harder task, and complex readout scheme, inspired us to expand from a one-crossbar system. Rather than using currents from the imprinted layer to solve a classification problem, we considered the case where those currents are passed forward, after being transformed at the hidden layer via an activation function, to a second crossbar. Therefore, we return to a system that looks more or less like the NoProp system introduced in Section 3.2.2, but with an additional level of complexity due to the pre-training/priming operation, conducted on the first layer weights W_{in}.

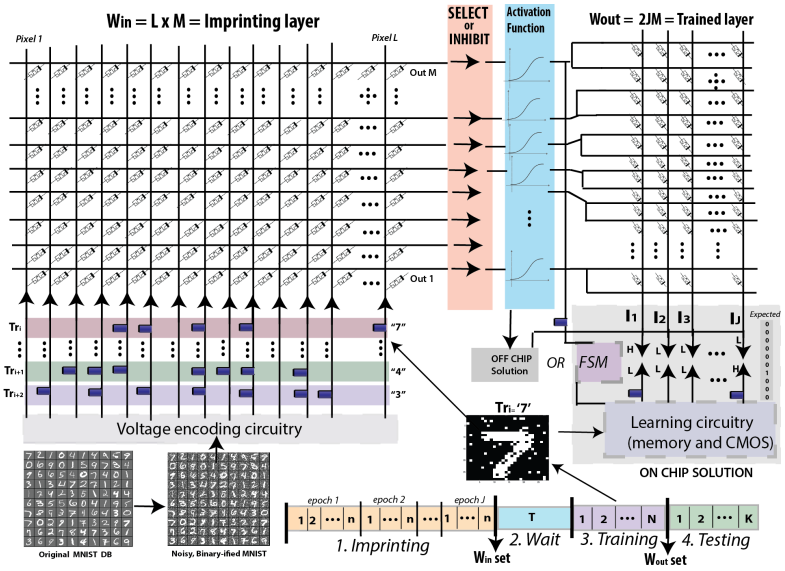


Figure 4.6: Conceptual architecture diagram for the dual-crossbar system that can compute higher dimensional tasks by projecting currents from examples to a second regression layer. One representative moment of the system - presentation of an image $Tr_i = 7$ during the training period- is depicted. The timeline below depicts operation through major phases.

4.2.5.1 System Description

Fig. 4.6 reveals a conceptual hardware implementation of our NoProp-improved system built with two crossbars of memristive devices, along with a timeline of its operation. As in the simple (one-layer) system, there are overall four phases: imprinting, waiting, training, and testing. In all cases, the original database is presented to the inputs of the first crossbar as binary, noisy, voltage vectors.

Options for first layer

We contrast two approaches: the random weights first-layer system (nanodevice No-Prop system introduced in Sec. 3.2.2) and a system which harnesses the STP to LTP transition to intelligently modify the first layer. In the former case, we use the NoProp system described in Sec.

3.2.2, with non-volatile nanosynapses placed randomly in the ON or OFF state.

In the latter case, the input crossbar uses ECM cells as described in the previous section. It is imprinted with images taken from the training dataset, using the same procedure as in Section 4.2.2. It is therefore the part of the system that uses the STP to LTP solution, and it yields a projection space W_{in} that will be used in training and testing. Imprinting happens epoch by epoch, with the total number of epochs identical to size of the hidden layer, M. A sequential programming could lead to timing bottleneck issues, as will be discussed later.

Hidden Layer Functionalities

Unlike the system detailed in Section 4.2.2, currents are not stored but fed to the the second stage of the system. Before they reach the second crossbar, they are passed through activation functions to increase dimensionality. For the random weights systems, we take difference of each pair of pre-synaptic currents from the hidden layer and use the $sign$ function as the activation function, as described in Section 3.2.2. For the imprinted system, the $tanh(\epsilon I)$ function was chosen since it can be easily implemented in CMOS [299] as well as engineered for variability. In our case, offsets were set randomly on each neuron and gain factor always $\epsilon = 10$.

Options to train second layer

In both cases, the second layer's weight matrix W_{out} acts as a regression layer, and is not imprinted or random, but trained. It does not exploit the STP to LTP transition of the nanodevices. Each input of the first crossbar is connected to two rows of the second crossbar to encode positive and negative weights, as already described in Sec 3.2.1.

A least squares solution may be obtained by collecting and computing weights at the end of training and importing them in a single shot (one-shot learning, or *ex-situ* mode), or may be computed iteratively as subsequent training examples are given (online learning, or *in-situ* mode). Ex-situ solutions require a pseudo-inverse operation, which might be complex to compute in hardware, or closed-form ridge regression, which may be easier to implement. Except for the results shown in presented Section 4.2.5.5, the results hereafter always collected and wrote weights in a one-shot fashion, using the closed-form ridge regression to obtain W_{out} given actual matrix A of training examples (composed of current vectors from W_{in} passed through the activation function), and expected matrix Y (composed of binary vectors of all labels):

$$W_{out} = YA^{\top} inv(AA^{\top}) \tag{4.5}$$

For the online learning (iterative) read-out, we always used the single example binary WH scheme detailed in Sections 1.4.5, 3.3.1, 3.3.2. Yet both single example and batch mode learning are compatible with generic architecture shown in Fig. 4.6.

4.2.5.2 Effect of Device Timing and Variability

Fig. 4.7(A) again shows that a sufficient number of patterns presented per epoch (imprinted neuron) $n > 20$ is a constraint for successful imprinting. Conversely, over-saturation is also

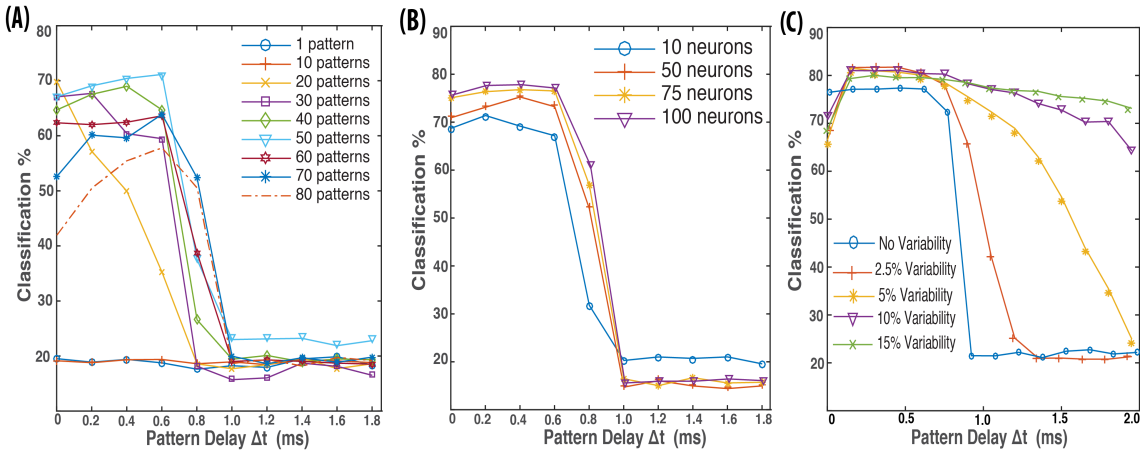


Figure 4.7: Classification rate as a function of delay between pattern presentation during the imprinting phase, for the ELM inspired system. (A) for different number of patterns presented per epoch (M=10) (B) for different number of hidden neurons M (n=50 patterns) (C) for different nanodevice variability cases (M=100, n=50). In every case, $T = 1s$, $N = 60k$, $K = 10k$, 10% noise applied to every image in imprinting, testing, training.

possible when $n > 50$ patterns are applied per epoch at the faster (smaller) time steps. In all uniform cases, performance drops when imprinting is too slow ($\Delta t > 1$ms). Fig. 4.7(B) shows that unlike the simple system, increasing output (hidden layer) neurons increases performance. This is due to the different random activations provided at each neuron. Whereas Fig. 4.4 showed a reduction in performance at increasing dispersion, Fig. 4.7(C) reveals the contrary case: maximum classification slightly increases. For instance, when $M = 100$ and at optimal wait, max 78% is reached in the uniform case compared to 82% for variable. While the uniform case shows a classification 'cliff' after 0.8ms, the 10%, 15% variable cases again show a broad tolerance to slower imprinting. One explanation is that, with high conductance evolution variance, some synapses are always excited enough to move from STP to LTP. This result is attractive, since nanodevice variability is transformed from a liability into a productive asset of the computing system.

4.2.5.3 Effect of Hidden Layer on Performance

Fig. 4.8 (a) shows that regardless of device uniformity or variability, imprinting W_{in} is demonstrably meaningful: imprinted systems substantially out-perform the ELM control cases at every value of M. This result can be compared with analogous priming of the first layer of ELM systems in software artificial neural networks. Such priming has already been reported to improve performance over the standard case [127, 297]. Here, we show a similar result subject to unique device timing constraints. Fig. 4.8 (a) also shows that imprinted systems with synaptic evolution variability consistently outperform the uniform case (where every synapse behaves identically). Nanodevice variability then allows for a greater variability between the hidden

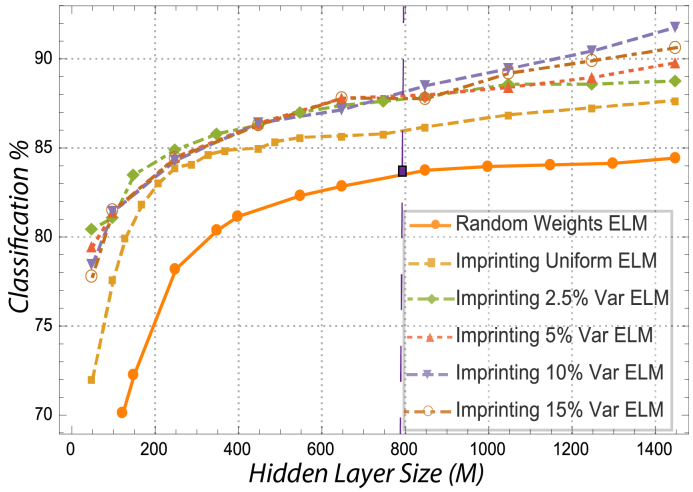


Figure 4.8: Classification rate as a function of hidden layer size M in different conditions: random weights on first layer (Random Weights ELM); imprinting on first layer, with no variability (Imprinting Uniform ELM); or variability (Imprinting Var ELM) on nanodevices. The single purple point/ dashed line at $L = M$ represents the direct regression solution obtained when all training images are presented directly to the second layer without any first layer (projections). In every imprinted case: $n = 50$ patterns are given per epoch (uniform), n=30 (variable cases), $T = 1s$ for burn-off period, $\Delta t = 200\mu s$, 10% noise is present in every imprinting, train, and test image, $N = 60k, K = 10k$.

neurons, enhancing the dimensionality of the data provided to the second layer beyond just the varying activation functions. At $M = 1450$, peak classification of 91.8% is obtained for variable imprinted systems, the uniform imprinted systems obtain 87.8%, and random weight ELM reaches 84.4%. At $M = L = 784$ (dashed vertical line in Fig. 4.8 (a)), random weight ELM performs similarly to a regression obtained by presenting all training samples directly to the second layer (83.5%). While the direct regression can only be made at $M = L$, to reach higher performances than standard regression, ELM requires substantially higher M values. Conversely, the rich underlying dynamics of nanodevices allows designers to do more with less in the imprinting cases.

4.2.5.4 Effect of Training Set on Performance

Whether in batch or online mode, minimizing training samples number N used to compose A can save energy and time. Fig. 4.9 shows classification rate as a function of training samples, at the case $M = L = 784$. At very low sample size, the rate of improvement is high; a steady state is reached around $N = 5,000$, and performance is already within 1-2% of maximum around $N = 10,000$. By $N = 2,500$, the variable imprinted system already outperforms the maximal result obtained for standard ELM (83.5%); uniform imprinting surpasses by $N = 8,500$. With the full training set, uniform and variable ECM projections ultimately reach new classification heights (87% and 90%, respectively).

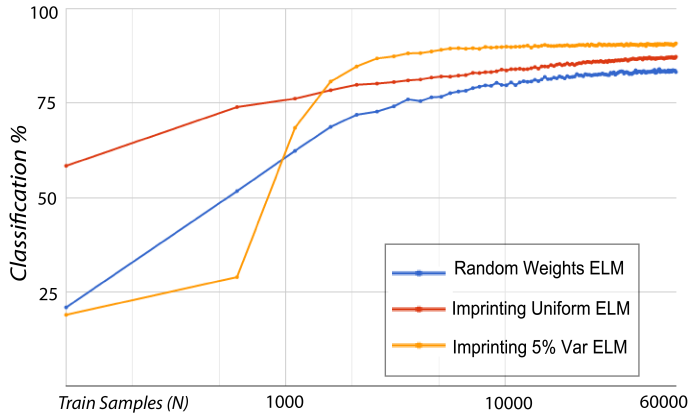


Figure 4.9: Classification rate as a function of number of training samples N used to compute W_{out} in the third stage, given $M = L = 784$ (the purple line/slice in Fig. 4.8 (a)), depicted on a log scale. Three two-layer systems cases are depicted: where W_{in} is set randomly with all ECM at low values (Random Weights ELM), and two imprinted systems where ECM cells are uniform and variable (5% dispersion), respectively. In every imprinted case: $n = 50$ patterns are given per epoch (uniform), n=30 (variable cases), $T = 1s$ for burn-off period, $\Delta t = 200\mu s$, 10% noise is present in every imprinting, train, and test image, $N = 60k$,$K = 10k$.

4.2.5.5 Comparison of Ex-Situ and In-Situ Learning

Next, we evaluate how online learning systems which implement on-chip readout using the parallel adaline/binary WH algorithm perform relative to the one-shot weight calculation/import approach. To test this possibility, we have integrated a non-volatile memristive device in the second layer (the volatile properties of the ECM cell would not be advantageous in this application). In particular, we used the TBFe device (Section 2.2) as a benchmark, and in the following sections always considered two variations on it. In the 'perfect' read-out device case, the devices are always uniform and do not suffer from asymmetry or other nonlinear effects. In 'worst-case', the device suffers both from inter-device variability ($\sigma = 30\%$) in its maximal/minimal values and typical conductance adaptation quantity ΔG; additionally, ΔG varies depending on the present conductance state affected (see 'Nonlinear' from Chapter 3 Section 3.1.1.3).

Figure 4.10, which shows the variation in performance as a function of hidden layer size M, again emphasizes that the imprinting approach boosts performance relative to the random weights approach case, and this advantage is particularly distinctive at very small layer sizes. Second, it emphasizes that the *in-situ* approach loses only $1-3\%$ in accuracy from the *ex-situ* perfect nanosynapse case but as much as $7-9\%$ in the non-ideal nanosynapse case at a large enough hidden layer size. At very small hidden layer size, the gap is more substantial. Third, the simulations suggest that defective nanosynapses have overall a more negative impact in the random weight online learning systems than the imprinted ones.

A more detailed analysis of how input noise and second layer variability in the read-out devices effects system performance was also conducted. As expected, continually degrading

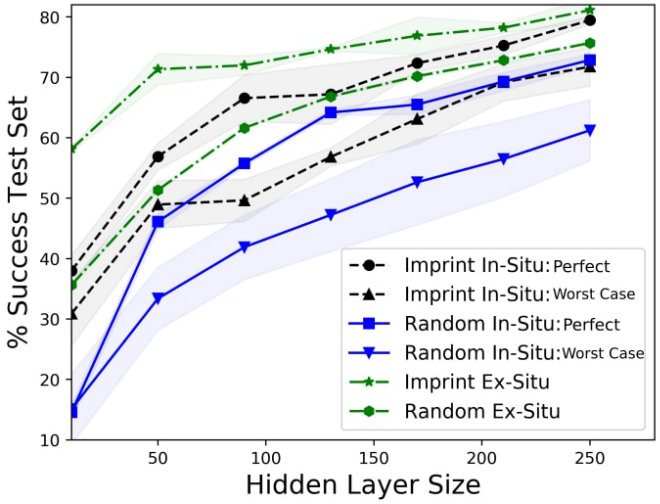


Figure 4.10: Four online learning cases, one for each first layer weight approach and with both qualities of read-out nanosynapses, are contrasted against the one-shot learning results already shown. In all cases, training set is a randomly shuffled version of the entire MNIST database presented twice, e.g. $N = 120k$, testing is the entire MNIST test database $K = 10k$. In every imprinted case: $n = 50$ patterns are given per epoch (perfect), n=30 (defect cases), $T = 1s$ for burn-off period, $\Delta t = 200\mu s$, 5% noise is present in every imprinting, train, and test image. Every point on the graph is a separate average of $m = 5$ total system trainings that have been performed; each system started with different random weights in W_{in} and W_{out} before imprinting and/or online learning, respectively.

the input images continues to decrease system performance (Fig. 4.11(a)). However, it seems to affect random weight systems slightly more than imprinted ones, and read-out synaptic arrays with imperfect devices more than perfect ones. While the latter result is intuitive, the second result probably relates to the ability of the imprinted system to naturally resist low amounts of noise in the burn-off or relax period, as demonstrated in Fig. 4.3, for instance.

As far as second-layer variability, we find that slight levels of variability between device maximal and minimal values and corresponding conductance evolutions (*e.g.*, 5-15 %) can actually slightly increase performance and assist the on-chip learning systems to more closely approximate the offline learning cases (written second layer weights), while substantial variability, especially greater than 20%, begins to substantially harm performance. These results are visible in Fig. 4.12.

4.2.5.6 Discussion

The simple system introduced in section 4.2.2 achieves promising classification on a simple task, and does so with minimal computing accessory as patterns are remembered naturally as a function of time and device properties. Explicit weight changes are not needed, which eliminates an impediment towards larger crossbars that require large circuit overhead for this

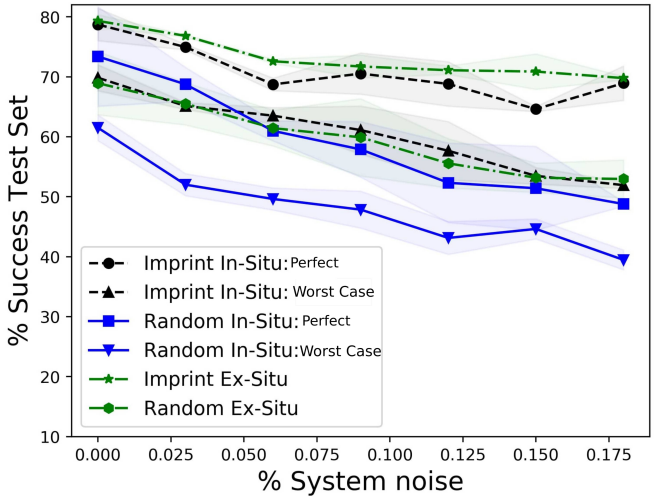


Figure 4.11: The six series demonstrate how four online learning cases and two offline learning cases respond to increasing corruption of the input images used to test and train the network. In the former four systems, one with each quality of nanosynapse (defective/imperfect or perfect) for each the randomly set ($W_{\text{in,rand}}$) and imprinted ($W_{\text{in,impr}}$) cases, where the W_{out} weights are always set using the crossbar-adapted binary WH learning scheme; in the latter two offline cases, W_{out} is externally obtained and written. In every system, $M = 250$, and every point on the graph is a separate average of $m = 5$ total system trainings that have been performed;

purpose. As memristor-CMOS systems have already been demonstrated to learn images of equivalent complexity [181], a physical implementation of our system is possible and could demonstrate further trade-offs. However, the readout involves a relatively complex procedure, and the system is sensitive to device variability. While it is a proof of concept for harnessing transition from STP to LTP in nanodevices, it has limited applicability to real nano-electronic system design.

Conversely, the imprinted ELM architecture introduced in section 4.2.5 is a promising lead for future nano-architectures. Imprinting a first layer with training examples definitively improves performance on the primary task. By achieving a far better classification at far smaller hidden layer size M than previously reported, the approach could dramatically reduce the total size, number of nanodevices, and CMOS neurons required to implement future ELM-inspired systems. Moreover, the fact that variable synapse ELM systems out-performed uniform synapse ELM systems is promising, as nanodevice variability is usually a serious concern. As nanodevices are naturally imperfect and structural synaptic diversity has been shown to enhance information coding in biological synapses [300], this implies that naturally variable filamentary nanodevices, such as our ECM cells, are excellent building blocks for future neuromorphic systems.

In general, this work represents a milestone for realizing a crossbar-friendly design that actively uses volatile memristive devices in a complex classification task. In [296], percentages of

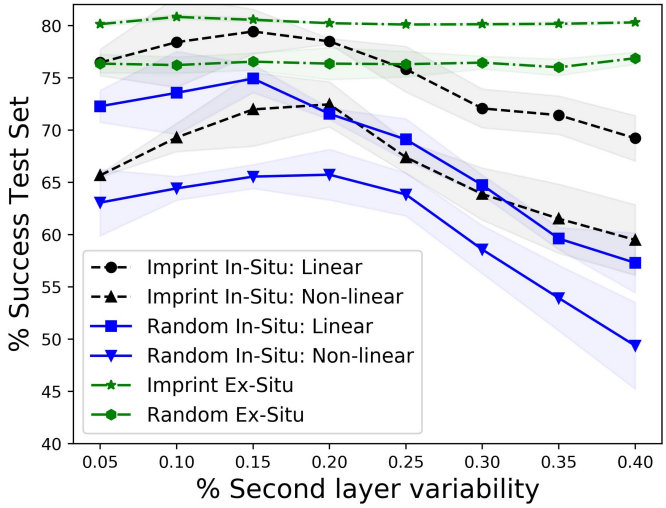


Figure 4.12: The six series demonstrate how four online learning cases and two offline learning cases respond to increasing global variability parameter applied to the second layer read-out weights. In the case of online learning, these imperfections directly affect the weight updates made throughout the adaptive process; in the case of offline learning, they still affect the write quality of the moment at which the off-chip weights are written. In every system, $M = 250$ and points are average of 5 total trainings.

80%+ on the primary task were only possible when the currents of several individual crossbars were combined and when two following layers (a multi-layer perceptron), provided the solution. Our proposed system reduces area, complexity, and performance in comparison to these past schemes.

By suggesting and briefly evaluating the impact of two different read-out strategies, we have uncovered important trade-offs. On one hand, since ridge regression solutions are iterative, *ex-situ* low sample (N) solutions to W_{out} can represent a trade-off between accuracy and speed/energy saving that might be intentionally exploited by approximate computing systems. We have also demonstrated that in very small systems one-shot training brings superior results. However, one-shot training requires a substantial overhead to store the projections from the first layer, and the circuitry to compute the matrix inverse on chip could be very large. For these reasons, especially in chips that benefit from a large enough hidden layer, the fully online learning approach is a performant and energy-efficient option.

Finally, while the imprinted systems broadly beat the control case of random weights, this approach can come with its own dis-advantages. Most importantly, this process takes a great deal of time due to a 'speed limit' set by device relaxation. If imprinting proceeds faster than $\Delta t = 100\mu S$, it oversaturates. Assuming $\Delta t = 200\mu s$, $n = 40$, $J = 10$, and $T = 1s$ (MNIST), then $T_{\text{tot}} = 1.1s$ is required to imprint W_{in}. While these delays may be a major liability in a commercial neuromorphic computing, the devices we studied were academic. Device engineering, in particular device scaling, could tune τ_{fac} to allow for faster pattern presentation, thereby nar-

rowing the gap between transient neuromorphic computing systems and non-volatile ones.

4.3 On-Chip Reservoir-Computing Inspired Scheme

In this section, we build upon the system introduced in Chapter 3, Section 3.2.2 by adding complexity at the hidden layer. This allows us to approximate some of the important advantages of the Reservoir Computing computing approach. As discussed in Chapter 1 Section 1.3.9, reservoir computing (RC) is a powerful computational framework because it approximates the full power of recurrent networks without requiring an onerous training procedure.

However, in contrast with RC systems, and more like the on-chip NoProp system shown previously, our system couples two crossbar arrays and operates in a feed-forward manner. In addition, it still uses random weights in the first crossbar, unlike the plasticity scheme demonstrated in Sec. 4.2. Here, we show that even with feed-forward operation, strong performance on tasks with time-dynamic can be achieved by exploiting time-dynamics within the hidden layer itself. This approach is reminiscent of a dendritically-inspired architecture proposed by Tapson [128], who used a variety of synaptic kernels to achieve promising results on spatio-temporal tasks. Ours uses simpler spatio-temporal integration schemes specifically designed for on-chip learning with emerging, highly analog nanosynapse models. Given the well-known advantages of the RC/NoProp paradigm (fast training, online operation) and the spatio-temporal data processing challenges inherent in an upcoming era of distributed computing, we expect this proposal should be of great interest to neuromorphic designers.

4.3.1 Architecture

4.3.1.1 Conceptual Depiction

A conventional RC architecture, visible in Fig 4.13 (a) is built with a sparsely connected graph of m excitatory and inhibitory neurons in the liquid, some or all of which are recurrent; this graph is excited along a set of input channels connected to W_{in} and feeding out along weights W_{out}, as was previously discussed in Section 1.3.9. As W_{in} is static, W_{out} can be solved as the pseudo-inverse of all collected output activations (A^+), given a matrix of labels or expected values (Z), as depicted in the bottom of Fig. 1(a).

Our proposed scheme, introduced in Fig 4.13 (b) also sets input weights and linearly regresses to achieve output weights, but now has a feed-forward fully-connected architecture to form a crossbar-compatible multi-layer learning system, as fully detailed in Section 3.2. To achieve the spatio-temporal features needed, the hidden neurons each possess a stateful value which corresponds to the past activations they have received within the present example. Sym-

bolically, given example s presented over k total time frames f, neuron m computes output as

$$O_{m,s} = sign(\sum_{f=1}^{k}\sum_{i=1}^{n} X_{i,f} W_{i,m}) \tag{4.6}$$

where n is the input dimensionality (total number of channels) and i is the index of the input channel. The $sign$ function means that, like an RC system, neurons have binary states: excitatory (+) or inhibitory (-). The scheme is built to perform well on tasks presented frame-by-frame such as audio, video, or other tasks mapped to a time domain.

Two variations of the architecture are possible:

- In the 'uniform' case, each hidden neuron integrates over every frame presented.
- In the 'sparse' case, each hidden neuron integrates only a subset of the time frames, specific to the hidden neuron: the sum over f must be limited to a subset of f values (hence, sparsity is enforced).

This is illustrated in Fig 4.13 (b), where the corresponding lettered hidden layer neurons integrate over the frame areas pictured below them.

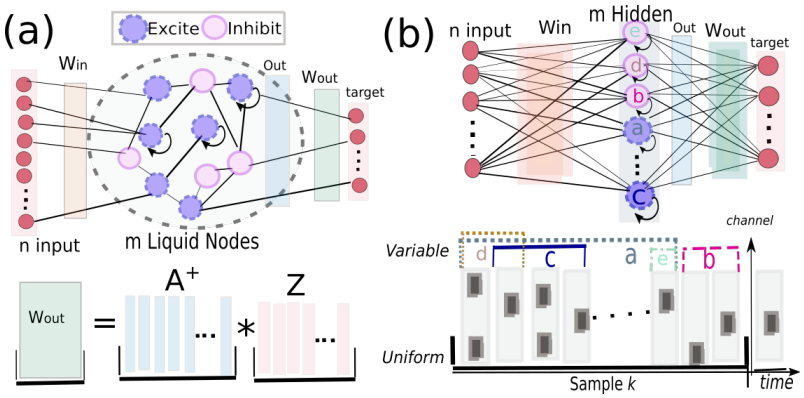


Figure 4.13: Conceptual illustrations of (a) the standard RC/LSM paradigm and the on-chip, (b) RC-inspired scheme proposed here.

4.3.1.2 Nanoelectric Implementation

Our architecture implementation with nanoelectronics is presented in Fig. 4.14. It is constituted by two nanodevices crossbars implementing a "projection layer" and a "readout layer" connected by circuits implementing the hidden neurons.

Input and projection layer

Inputs are presented to the first layer as consecutive frames of voltage pulses to n input channels. Weights in this layer are set randomly at the initialization of operation and unchanged. Two modes were considered: analog and spiking. Following dissemination through the crossbar, output currents at the other side are subtracted in pairs (so as to achieve negative weights)

and translated through standard op-amp circuitry into the voltage domain. Only the binary value of this voltage, or its sign are relevant to the following cell.

Hidden neurons activation

Different technological options are possible for implementing the time-integration performed by hidden neurons circuits. It can be achieved digitally using counters, or in an analog fashion using capacitors, as is often done in the neuromorphic electronics field [30]. It could also potentially performed using analog nanosynapses. As mentioned above, each hidden neuron integrates activation over either all frames (uniform scheme) or smaller, random subsets of frames (variable scheme). To realize the variable scheme a small memory needs to be associated and configured with each hidden neuron. In addition to standard memory components, another crossbar of nanosynapses set randomly on or off could be used to constitute the physical reference for a sparsity matrix; in this case, dot-product operations between layers of a 3D crossbar array would significantly reduce required overhead.

Readout layer

The readout layer uses the weight space of pairs of analog nanosynapses to provide on-chip regression corresponding to the activations from the hidden neurons. The scheme proposed for the readout layer is the same as the parallel on-chip regression system already described in Section 3.2.1, and in principle held constant relative to the standard nanodevice NoProp system of Sec. 3.2.2.

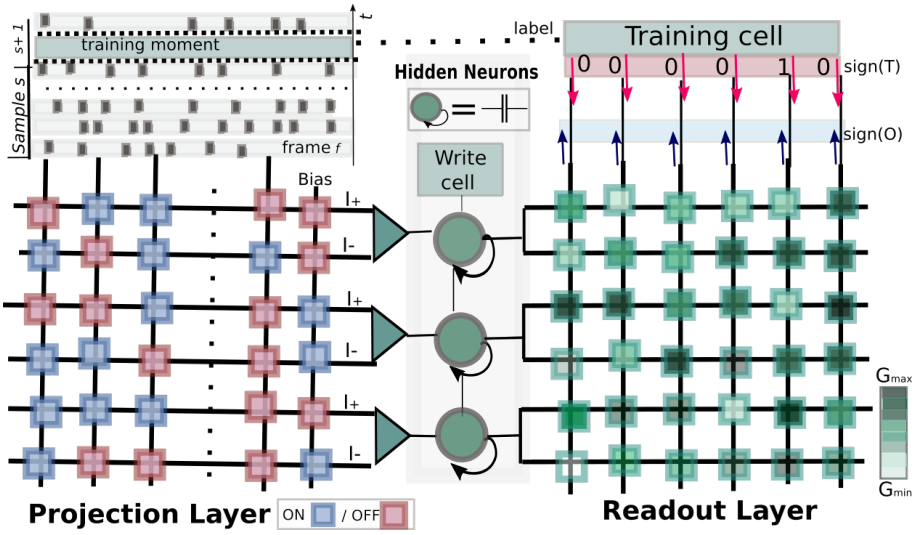


Figure 4.14: Schematic illustrating the simulated nano-system used to obtain results. Each square icon in both cartoons represents a nano-synapse. In the first layer (projection) either binary device and/or natural dispersion around the maximal or minimal values of the device can be used; in the second layer (readout), the full weight range is exploited in order to achieve on-chip regression using a simplified version of the Widrow-Hoff or delta learning rule.

4.3.1.3 Nanosynapse

We used a realistic nanosynapse model extrapolated from experimental data obtained from the TBFe device(Section 2.2), and that actively accounts for inter-device variability ('Imperfect' from Chapter 3 Section 3.1.1.3, or Chapter 2 2.5.5.1).

For the projection layer (W_{in}) which is unadaptive, the devices were either in 'binary mode' and programmed randomly at one extrema (perfect devices); or were all programmed randomly around the same extrema (ON or OFF), taking advantage of the natural Gaussian curve as proposed in [263] (imperfect devices). For the adaptive read-out layer (W_{out}), we always used nanosynapses with 7 bits or $g = 128$ addressable states.

4.3.2 Considered Tasks and Methodology

First, we considered a sub-set of 500 examples from the TI-46 corpus of spoken digits. These spoken digits were encoded using an artificial cochleagram; pre-processing was based on the passive ear model or filter bank designed by R.F. Lyon [301]. Full details on this pre-processing and database are available in [302]. The processed data we used in our tests shows 77 channels with between 50 and 100 time frames or steps during each example digit. The dataset was imported from and used in the context of a generic software simulator for reservoir computing written in Python (Oger) [161]. The identical database and pre-processing methods were used to present this set as training and testing examples to a custom discrete-time crossbar simulator which tracks the evolution of all hidden neurons activation values and (second layer) device weights; this software was also written in Python.

The 500 samples were divided into a training set of 350 samples, and a test set of 150 samples. The training set is labeled and used either to teach the readout layer or to build an output matrix for ridge regression; conversely, testing examples are presented label-less, and the prediction or inference is compared externally to the system. Each full training procedure consisted of 10 subsequent presentations (epochs) of the sample set (5k iterations); examples were always shuffled between each epoch.

Although this task has intrinsic time dynamic, the number of total samples is very small. In addition, it is a highly separable task, and therefore somewhat trivial to solve; as demonstrated in Fig. 4.15 (b), the data-set resolves into relatively separable clusters (classes). Therefore, to further estimate the generalization abilities of our architectures, we also tested with the well-known and more difficult MNIST database of handwritten digits [159], which consists of 70,000 samples (10,000 tests, 60,000 training examples) and is known to be non-linearly separable (also see, Fig. 3.1, (e)). In order to lend time-dynamic to the task we presented all individual examples (from training and test) as 28 frames presented subsequently to the input layer, as in [296]. As in the TI-46 database, training examples were always shuffled (presented in a different order). Training with this database, due to its large size, consists of only a single epoch; no distortions or pre-processing were made before presentation; reported scores are always based

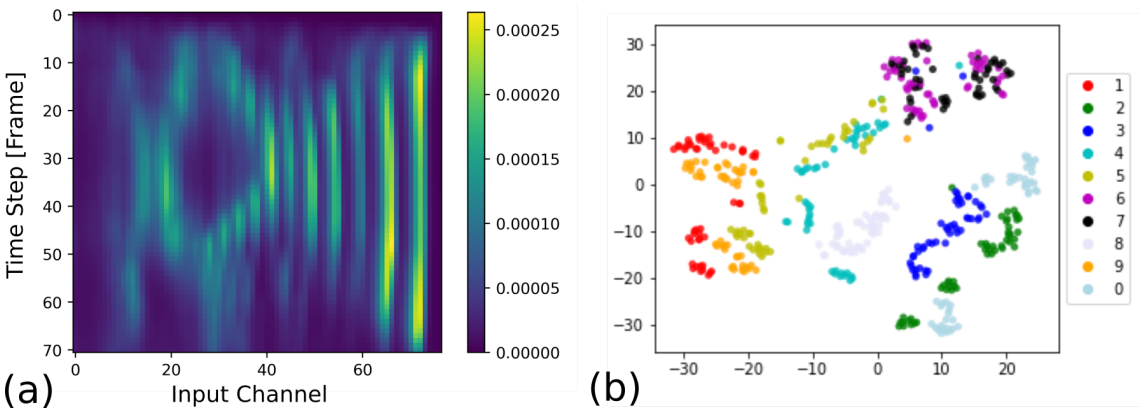


Figure 4.15: (a) Shows, in matrix form, one training example from the Lyon task- this one of a spoken version 'three'. As visible, the example is composed of many consecutive frames in analog mode. The values have been interpreted from their original values to voltage value for presentation to the learning network. (b) Shows a clustering visualization of 350 examples (entire training set)- each one such as (a)- in a dimension-less space produced by the tSNE algorithm.

on classification on the entire test database.

For analog mode presentation: pixel intensity or channel's analog value for cochleagram were input directly as the corresponding voltage pulses. For spiking mode, all negative cochleagram values (range $[-1,1]$) were converted to 0 (no spike), and all positive to 1 (spike); for MNIST, pixel luminosity greater than 0.5 spiked, and that below did not (range $[0,1]$). When applying noise to spike channels, the noted percent of pixels or channels at each moment flipped from one bit value to the other (on average).

4.3.3 Learning Performances

4.3.3.1 Effect of Hidden Layer size

As visible in Fig. 4.16, both the software reservoir and our proposed crossbar interpretation are strongly dependent on a large enough number of nodes in the reservoir or hidden layer to reach robust performance. However, while the software reservoir achieves > 95% already by $M = 100$ neurons, a dimensionality of at least $M = 200$ is required for the crossbar based version when it is built with perfect nanosynapses. As visible, the sparse sampling method creates a noticeable but not dramatic improvement.

The case of the MNIST challenge shows the same strong dependence of performance on hidden layer size, but in this case, the performance improvement between the uniform integration and sparse schemes is dramatic. The uniform case only approaches 80% at large number of hidden neurons; this is an inferior result for a two layer system, given that a one-layer crossbar learning system using the same device can already achieve this result (Section 3.3.1). This suggests that the simple, or uniform integration scheme that was used earlier is not adequate

to differentiate between many presented MNIST samples in the training phase. Two factors may be at play: first, when artificially converted into the time domain few input channels are available in general; second, integration (summation) across the entire time slice cannot enhance dimensionality but only reduce it. This would be especially punishing in the MNIST case due to the complexity of the training data (which is not, in general, linearly separable). In contrast, enforcing hidden layer sparsity results in a substantial boost in performance, with ultimate performance at 95% at large hidden size. To obtain these results a random subset of one fourth of the frames from each image was selected by each neuron; these sampling windows are consistent over all presentations and between training and testing (the tuning curve is constant).

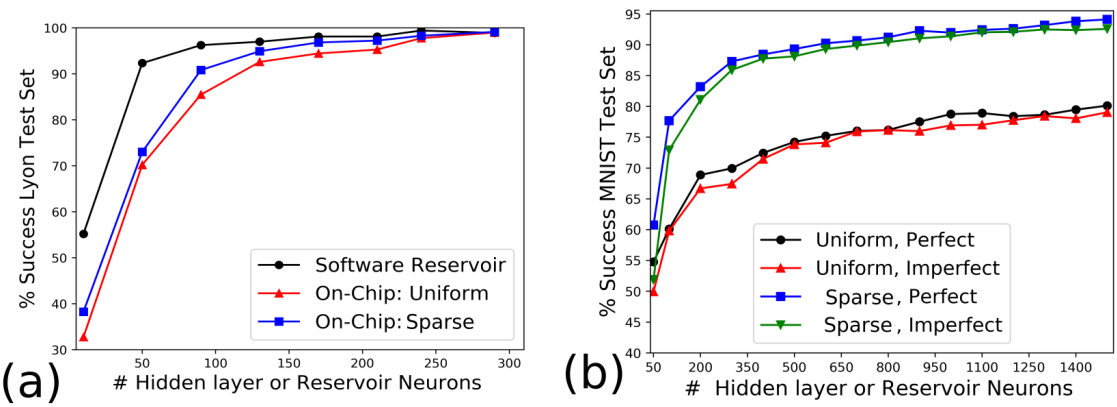


Figure 4.16: Performance of the considered architecture on the two chosen tasks. (a) shows performance on Lyon for software, on-chip simple integration, and on-chip sparse integration; (b) shows both considered integration schemes for the MNIST challenge where mild variability ($\sigma = 10\%$) is showed in the 'imperfect' series

4.3.3.2 Effect of degree of sparsity

Next, we explored whether optimal sparse connection levels exist. As visible in Fig. 4.17 (a), the optimal sparsity greatly depends upon the task; the optimal sparsity for MNIST is quite high, with best performances coming when between 40-20% of frames are active per neuron. For the Lyon task, 85-65% connection produces the best results. While the outcome for systems performing the MNIST task with variable devices ($\sigma = 10\%$) is mostly equivalent to those systems performing it with ideal devices, the same cannot be said on the Lyon task. In this case, systems learning with imperfect devices increasingly trail the systems learning with perfect ones as sparsity increases.

4.3.3.3 Effect of readout device variability

While projection layer variability can easily be taken in stride by using normal distributions as an asset, imperfect devices in the adaptive (readout) layer can confound the learning rule's

performance. Fig. 4.17 (b) shows performance on each of the two considered challenges as a function of increasing standard deviation or dispersion of maximal, minimal conductances and thresholds at $M = 200$ for the Lyon task, and $M = 1200$ for the MNIST task. As visible, slight variability is not very harmful, while greater variability starts to substantively effect performance. Notably, the rate of deterioration on the Lyon task is much higher than the MNIST task; in the far smaller readout crossbar for the Lyon task, each device is more important to successfully resolving the problem.

The impact of variability in the second layer is further clarified in Fig. 4.18, which shows the case of solving the Lyon task with the sparse, variable sampling windows. As visible, the on-chip algorithm can only mimic software results perfectly with uniform devices, but 5% and 10% serve as close approximations, due to the adaptive nature of the algorithm. However, as variability is increased to above 10%, both the average performance and the standard deviation of error increases. Reducing the impact of device imperfections in general could be realized via better engineering or through architectural tweaks such as using more than two nanodevices per single nanosynapse element. A few engineering techniques to achieve this have been listed in [260].

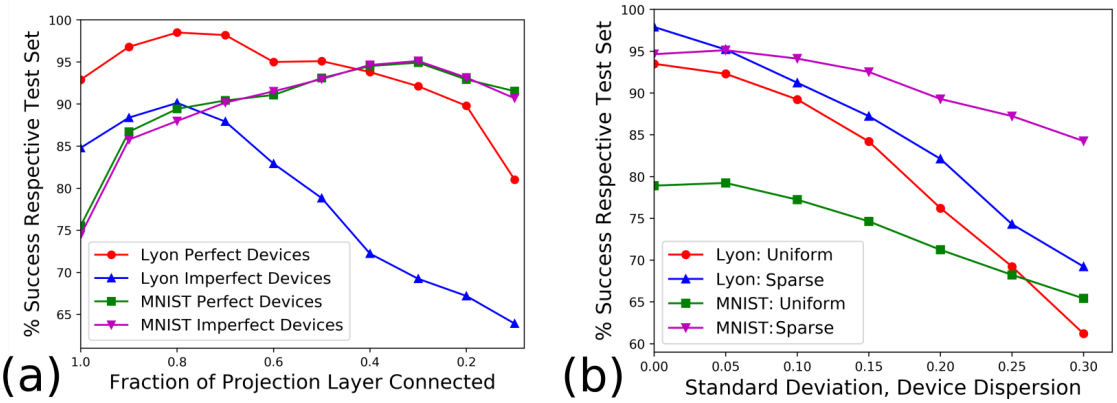


Figure 4.17: (a) shows response of variably integrating systems to the degree of spareness in the first layer, where 1.0 is the default (uniform) case; (b) shows deterioration in performance as dispersion the G_{on}, G_{off} parameters is increased

4.3.3.4 Fast Training Capabilities

One of the signature benefits of the RC/ELM paradigm is that such systems generally require only a fraction of the quantity of training examples needed to solve the problem in more computationally demanding methods; *e.g.* multi-layer backpropagation or convolutional neural networks often require dozens of epochs (millions of individual updates) to properly converge. As illustrated in Fig. 4.19 (a), the Lyon and MNIST tasks achieve within 10% of the maximum performance in as few as 20% (12k) training examples, in the former case, and 60% (2.1k) training examples, in the latter case. For even more approximate applications (less stringent ac-

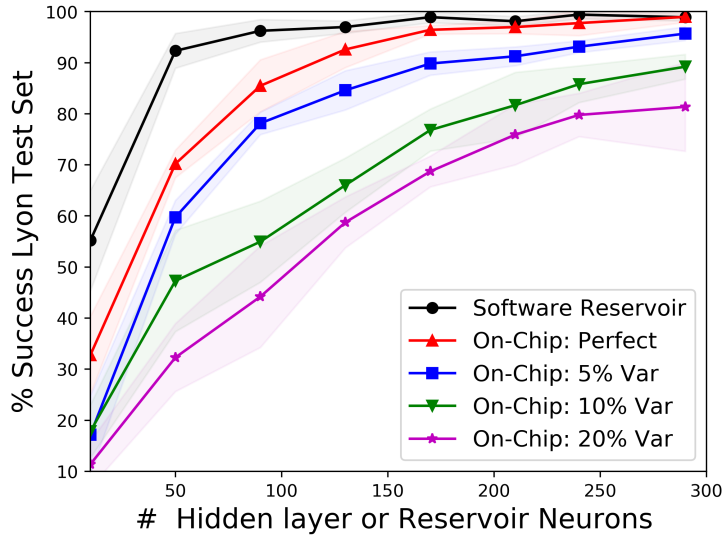


Figure 4.18: For solving the Lyon task with the variable hidden layer, progressively larger systems are grown and tested, given a set of examples for how non-standard the second-layer readout crossbar is (each colored series). Every point is the mean of 10 different simulated systems with a different random starting set of W_{in} and W_{out} values clipped between the devices' extrema, with the shaded region around the lines demonstrating continuous error 'bars'.

curacy requirements), 80% of performance can be reached in only 5k samples, for the case of the MNIST, and 1.4k samples, for the Lyon case. As small error rates lead to substantial energy savings, these results suggest the system is an attractive approximate computer.

4.3.3.5 Performance with Spiking Input

In neuromorphic applications, spike coding bestows extreme energy efficiency [303]. We tested the capability of our system to still provide strong solutions when the problem was presented in either in perfectly presented spike form, or imperfect (increasingly noisy) spiking channel input, as described previously. As visible in Fig. 4.19 (b), the initial conversion from analog to digital (spiking) presentation has a slim effect on ultimate performance (between 1.1% and 0.4% drop from ideal analog case). With increasing noise, performance deteriorates slowly; on a slightly noisy channel (5% bit flips) performance drops only 4-8%, and even with nearly one third of the channel corrupted performance is still within 80% of the maximum. The system's performance degradation on the Lyon task is less than that on the MNIST challenge; this shows that encoding the latter on only 28 channels is intrinsically difficult and with added noise many of the features are lost.

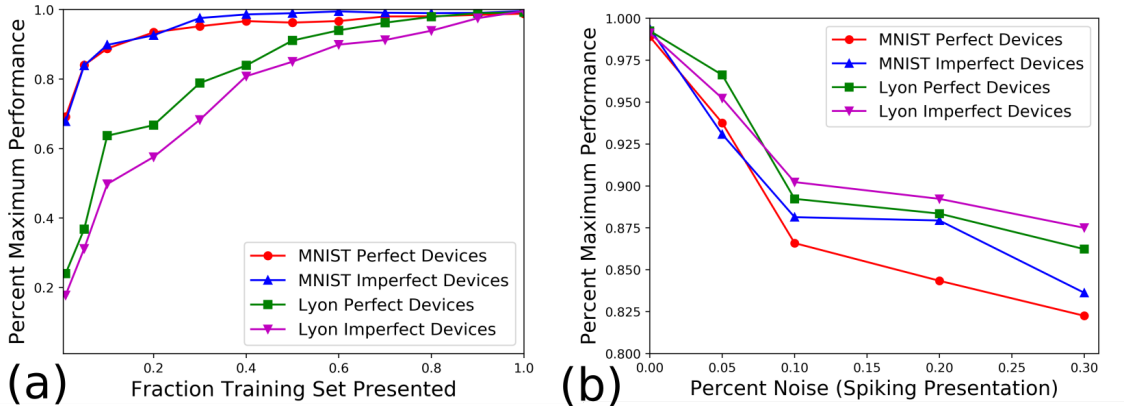


Figure 4.19: (a) shows convergence speed of the considered systems as a fraction of the total given samples in other graphs; (b) shows degradation in performance as the input channels to the first layer are increasingly simplified (analog to digital) and corrupted (noise).

Device/Task	Full Performance	10% Loss	20% Loss
ENODe-Lyon	0.65mJ	0.39mJ	0.182mJ
TBFe-Lyon	154mJ	92.4mJ	43.12mJ
TBFe-MNIST	11.09J	2.22J	0.924J
ENODe-MNIST	46.8 mJ	9.36mJ	3.9mJ

Table 4.2: Programming cost for two possible nanosynapses

4.3.3.6 Energy Estimates

Like the nanodevice NoProp systems, this scheme keeps the first layer weights static and trains the second layer using a conditional programming pulse scheme (not every output neuron is programmed at every training step). Due to high voltage levels for the simulated TBFe polymeric device ($V_p = 4.4V$), we also include energy estimates using an alternate polymeric electrochemical nanosynapse called ENODe [273] where $V_p = 0.5mV$. The energy cost per elementary write operation for the two devices is computed finally as $0.077\mu J$ and $0.325nJ$, respectively. Total programming expenditure for the Lyon task and the MNIST task for both polymeric nanosynapses at full training (full performance), within 10 % of max performance, and within 20% of max performance following the analysis in Fig. 4.19 (a) are visible in Table 4.2; system (hidden layer) size for Lyon is $M = 200$, and $M = 1200$ for MNIST. While this estimation is only raw programming cost, power draw for sum/ integration operations in the hidden layer, and checking/routing of error should be far less, especially if they use nanosynapses.

4.3.3.7 Benchmarks

Our best obtained results on the Lyon task were very promising; they emulate not only RC software results, but the best obtained results with an RNN approach (Long-short term memory cell (LSTM) networks) [304]. To test how critical the two layer design is to the high performance we achieved, we simulated a one-layer system with the training data directly presented (only the readout layer). Using this approach, we achieved only 56% using the uniform and 62% using the variable spatio-temporal integration scheme. As suggested in [128], the critical operation expressed by synaptic (in our system, hidden layer) kernels is to project inputs into a higher dimensional space.

In comparison to the results obtained in [296], which again presented that database similarly to memristive crossbars, we obtained superior results while also avoiding off-chip computational requirements. At the given hidden layer/reservoir size ($M = 1500$), top obtained result is superior to both the standard NoProp and imprinted first-layer NoProp systems by 3% and 5% on the M-NIST test set, respectively.

4.3.4 Discussion

While the proposed design has strong similarities to the RC paradigm: the first layer weights (synaptic attractions) are fixed, standard regression can be used to extract the value of this processing network, and neurons show both positive (excitatory) and negative (inhibitory) activations, like ELM, the system is feedforward, not recurrent. As recurrence bestows a higher memory capacity on a reservoir, it is not surprising that more hidden neurons are required in our case than a true reservoir. The computational capabilities of RCs and ELMs have previously been compared in [306], which noted that while RCs imply an unfortunate trade-off between non-linearity (mapping performance) and memory capacity (ability to retain said map beyond the fading time window), a synthesis of ELM and RC approaches might overcome this dilemma. Our system implicitly addresses this trade-off by forcing non-linearity at a large set of hidden neurons, and preserving capacity through stateful variables.

Our results with the frame-by-frame MNIST task suggest that achieving strong performance on a high-dimensional task requires more than simple spatio-temporal integration. When hidden neurons time variability is included, *e.g.* diverse activation functions, the computational power of the system amplifies as neurons specialize in their preferred frames from the database. This approach forces greater sparsity (breaks all-to-all connectivity), and closely resembles the concept of tuning curves used in [100] to implement efficient neural coding.

4.4 Conclusion and Perspective

Both of the considered improvements to the typical NoProp systems demonstrated major benefits. In the case of the imprinted systems obtained as a result of the plasticity transition occur-

ring within the silver ionic nanodevices, there were several critical benefits:

- All other things equal, *i.e* at equivalent hidden layer size M and given the same number of training examples, multi-layer plastic systems with the imprinted first layer always performed better on the test set of the given benchmark task (M-NIST) (Figs. 4.9, 4.8.)
- Imprinted systems benefited from variability in the first-layer devices used for the projection, (Figs. 4.5, 4.9, 4.8.) First layer variability would have no benefit in standard NoProp systems.
- Imprinted systems intrinsically benefit from greater noise immunity.
- Imprinted systems learn more efficiently in the online learning mode than random weight systems learning in this mode (Figs. 4.10, 4.12.)

However, the imprinting process requires time to present the images and to allows the devices to relax. Worse, while relaxation time T is a constant, since the total imprint time scales with the number of hidden layer neurons, $T_{\text{impr}} = MT_j$, this method could become particularly onerous for very large systems. A key improvement to this system and work would be a more efficient way of pre-computing or priming weights in parallel in a large crossbar environment.

The reservoir-inspired system, which added complexity at the hidden layer to improve results over a standard nanodevice NoProp system, indeed achieved this goal, but only when sparsity was enforced by creating variable sampling windows at the hidden layer neurons. The system approached perfect performance on the Lyon spoken digits task, and by approaching a performance of 95% at hidden layer size of $M = 1400$, the system outperformed a NoProp system with twice the hidden layer size (see Table 3.1). Even more dramatically, note that the dimension or overhead of the first layer was actually *28 times* less, since the MNIST task was presented slice-by-slice in order to realize the artificial timing effects. In addition to these accuracy and reduced overhead benefits, the system learned even more quickly than the standard NoProp system, converging within 90% of maximal performance in less than one-third of a full epoch (full presentation of the training set; see Fig. 4.19 (a)) . Fast convergence additionally means that the RC-inspired system can also save about 3-5x as much relative energy relative to the standard NoProp system in approximate computing applications.

In the future, more concrete hidden layer designs should be specified, especially those exploiting memristive devices due to their low area overhead. As is, the final energy and overhead benefits of the system cannot be finalized due to the variety of competing designs. As far as improving performance even further, inter-neuron effects could be explored to imbue the system with even more power to tackle complex non-linear or stochastic effects. Two possible designs to inter-connect hidden layer neurons for greater computational power are the time-delayed reservoir (TDR) approach [129] or the simple cycle reservoir approach [308].

Conclusions and future work

"Sometimes it seems as though each new step towards AI (Artificial Intelligence), rather than producing something which everyone agrees is real intelligence, merely reveals what real intelligence is not. "

Douglas R. Hofstadter

"IN *the context of neuromorphic engineering, it might seem that a panoply of algorithms, concepts, and physical devices are available, and so that picking any one of them , or any combination of them, is difficult. However, in this work care has been taken to emphasize, whenever possible, concrete models and devices over abstract ones. Given this lens, the organic memristor device introduced and studied, though far from perfect in terms of energy and scaling properties, serves as a great test case for future on-chip learning systems. By building and extending this model to new domains, and by testing the complementary performance of other and slightly more exotic devices with intrinsic time-dynamic features, the ultimate objective of the thesis- to explore the design space of future neuro-computers- has been realized. In turn, several designs and directions towards this ultimate goal are now under development. "*

THIS concluding chapter quickly recaps the significance and summary of this work, before discussing some natural extensions of the research.

Summary

In the next era of distributed computing, brain-inspired inference engines that perform data science operations locally rather than in distant servers would be a major advantage, from the perspective of reducing energy costs, reducing environmental impact, and increasing security and ownership of local computing and/or data resources. A new generation of emerging non-volatile memory devices are a leading candidate to achieve this neuromorphic vision. Using both theoretical and experimental works, we have shown that efficient physical realization of modern artificial neural network (ANN) architectures using emerging memory devices (memristive nanodevices) ought to whenever possible maximize the intrinsic (analog, temporal) abilities of these devices, while also being mindful of their physical limitations. To that end, a major consideration has been the design or optimization of architectures that resist or even exploit these natural limitations (variability, non-linearity, etc).

Three major lines of exploration have supported this theme. First, a new variety of polymeric redox memory device was characterized and simple as well as complex compact models were developed to assess its unique properties in a circuit environment. Importantly, we theoretically and then experimentally showed the compatibility of this exciting new device with a key learning algorithm that allows for scalable computing in the ultra-dense crossbar environment. In our experimental work, our organic devices successfully and automatically adapted themselves as reconfigurable logic gates by cooperating with conventional circuitry and a field programmable gate array (FGPA). In addition, we imagined and implemented far larger simulated learning experiments with this device, including notably with the M-NIST handwritten digits database. In the context of these tasks, critical device limitations such as asymmetric device behavior between conductance modes (SET, or increasing conductance, and RESET, or decreasing conductance) posed a core limitation, while typical inter-device parameter variation was handled far better.

Second, we abstracted away from this particular device and considered a variety of multilayer memristive neural network systems. In particular, we developed and simulated variants of random projection (NoProp) and backpropagation (MLP) learning systems on a suite of tasks. These local learning systems again showed critical dependencies on physical device constraints; notably, analog richness and non-linearity. By exhaustively considering these parameters, we uncovered non-obvious trade-offs between the efficiency of these learning algorithms in the context of emerging non-volatile memory learning systems. Notably, our results highlighted that random projection systems learn quickly compared to systems learning with back-propagation- which saves time and energy- and can learn with less perfect nanosynapses- a major asset in a pre-industrial context.

Lastly, we examined how standard feed-forward ANNs designs may be modified to exploit temporal effects, either at the device or system level. In this context, we focused in particular upon improving the bio-inspiration and performance of the NoProp system. We considered two such major improvements. In the first case, we enhanced the performance of standard random-weight architectures with subtle plasticity effects in the first-layer; these effects were obtained using a silver ionic nanodevice with an intrinsic plasticity transition behavior. In the second, we improved the intelligence of the hidden layer sub-system in order to increase the accuracy and speed of the on-chip learning system. In all of these improved systems, we always considered to some extent what the impact of non-ideal effects such as inter-device variability, corrupted input channels (noise), and others. Depending on the proposed system, they were either immune, very, or somewhat resilient to these effects.

Future Work

Three lines of possible future inquiry stem out from the works completed here; one in terms of device optimization and scaling with the nanodevice studied most closely (organic memristor), one in terms of general optimal learning methods for on-chip hardware, and one in terms of architectural/systems strategies in fully scaled synaptic grid computers.

In order to optimize the new class of organic memristive device used in the small system learning for scale in larger systems, device engineering as well as systems-level mitigation strategies to rectify the asymmetry between SET and RESTET issues encountered would be a critical task. As demonstrated in both the experimental and theoretical work, this was a critical constraint to effective on-chip learning; while an alternate learning mode does exist taking this into account (SET only mode, or continuous SET followed by ocassional RESET), it does not benefit from many of the efficiencies that would be critical at scale and which only comes when using SET/RESET together . Further device characterizations and simulations could help reveal to what extent the RESET operation in the polymeric device involves violent destruction (burning) of filaments, and if this is the case, possible changes in the polymer's chemical composition be implemented to test if a more gradual RESET mode is possible. Otherwise, special circuits and programming strategies may be considered to rectify this issue. Somewhat analogously to the case of Phase Change Materials, which require special circuit to carefully and extremely quickly heat the device between the crystalline and amorphous stage, perhaps very short programming pulses are required for gentle conductance decreases.

From the perspective of machine learning and optimal learning approaches, in the future it would be intriguing to expand the supervised learning approach of this thesis to additionally include semi-supervised and un-supervised approaches. As plasticity effects can implement un-supervised learning operations, these effects could also be explored with nanodevices. Ultimately, in the future we hope to further consolidate supervised decoding systems with more complex, un-supervised encoding layers, as suggested in [309]. In particular, enhancing the

multi-layer plasticity transition system into a completely, or semi-supervised learning system, would be an achievable and interesting extension of this work.

From the perspective of building scaled architectures with emerging ReRAM devices, it would be interesting to compare the rich analog approaches considered here with the complex binary adaptations of DCNNs have recently been proposed for on-chip implementation with resistive memory devices [310–312]. Usually, these assume the use of binary devices or use of complex, analog devices in 'binary-mode'. While such designs simplify many of the operations discussed herein by greatly reducing the weight range of their constituent nanodevices, they must then also store highly analog gradient updates ([268]). As a future set of work, it would be interesting to analyze which trade-offs, in terms of energy footprint, speed, overhead, etc, exist between larger networks using far more binary devices and those exploiting analog and temporal aspects of nanodevices more fully as we have attempted to do here.

Lastly, while in this thesis we have mostly considered standard machine learning audio and/or vision classification tasks, in the future it would be interesting to adapt the nanodevice ANNs we have proposed here to function on more realistic, open-ended data stream environments, such as streaming video. In this context, ANNs can achieve functions other than simply classification as well. In Sec. C.1, we have proposed an early version of an anomaly/novelty detection system built with memristive nanodevices. In this system, the objective is not to reveal what an unknown class is, but to compute an anomaly or outlier status relative to a baseline. In the future, we hope to describe and design nanosystems that integrate both the essential classification/regression activities shown here, alongside more neuro-inspired pre-processing filters, *i.e* for anomaly, associative memory, or sensor fusion purposes.

List of publications

Peer-Reviewed Journal Articles

✠ **C.H. BENNETT**, JACQUES-OLIVIER KLEIN and DAMIEN QUERLIOZ, "Sensitivity of non-volatile memory neural networks to device constraints" , *Frontiers in Neuroscience*, 2017. *In Preparation.*

✠ **C.H. BENNETT**, JEAN-ETIENNE LORIVAL, FRANCOIS MARC, THÉO CABARET, BRUNO JOUSSELME, VINCENT DERYCKE, JACQUES-OLIVIER KLEIN and CRISTELL MANEUX, "An Organic Memristor Compact Model Dedicated to Neuroinspired Circuits" , *IEEE Transactions on Multiscale Computing*, 2017. *Accepted/In Press.*

✠ YU-PU LIN[†], **C.H. BENNETT**[†], THÉO CABARET , DAMIR VODENICAREVIC, DJAAFAR CHABI, DAMIEN QUERLIOZ, BRUNO JOUSSELME, VINCENT DERYCKE and JACQUES-OLIVIER KLEIN, "Physical Realization of a Supervised Learning System Built with Organic Memristive Synapses" , *Scientific Reports*, vol. 6, No. 31932, 2016. [†]: co-first authors.
doi:10.1038/srep31932

✠ DJAAFAR CHABI, ZHAOHAO WANG, **C.H. BENNETT**,JACQUES-OLIVIER KLEIN and WEISHENG ZHAO, "Ultrahigh density memristor neural crossbar for on-chip supervised learning." , *IEEE Transactions on Nanotechnology*, vol. 14, No. 6, p. 954 - 962, 2015.
doi:10.1109/TNANO.2015.2448554

Peer-Reviewed Conference Articles

✠ **C.H. BENNETT**, DAMIEN QUERLIOZ and JACQUES-OLIVIER KLEIN, "Spatio-temporal Learning with Arrays of Analog Nanosynapses" , *Proceedings of the IEEE*, 2017.

✠ **C.H. BENNETT**, DAMIEN QUERLIOZ and JACQUES-OLIVIER KLEIN, "A recurrent crossbar of memristive nanodevices implements online novelty detection" , *Proceedings of the IEEE*, p. 1-4, 2016.
doi:10.1109/ICRC.2016.7738689

✠ **C.H. BENNETT**,SELINA LA BARBERA, ADRIEN F. VINCENT, JACQUES-OLIVIER KLEIN, FABIEN ALIBART and DAMIEN QUERLIOZ, "Exploiting the Short-term to Long-term Plasticity Transition

in Memristive Nanodevice Learning Architectures" , *Proceedings of the IEEE*, 2016.
doi:10.1109/IJCNN.2016.7727300

✠ **C.H. BENNETT**, DJAAFAR CHABI, THEO CABARET, BRUNO JOUSSELME, VINCENT DERYCKE, DAMIEN QUERLIOZ and JACQUES-OLIVIER KLEIN, "Supervised Learning with Organic Memristor Devices and Prospects for Neural Crossbar Arrays" , *Proceedings of the IEEE*, p. 181-186, 2015.
doi:10.1109/NANOARCH.2015.7180609

Book Chapters

✠ **CHRISTOPHER H. BENNETT**, ALDO JESORKA, GORAN WENDIN, and ZORAN KONKOLI, "On the inverse pattern recognition problem in the context of the time-series data processing with memristor networks" , *Advances in Unconventional Computing, Springer*, p. 735-757, 2016.

Presentations and Posters

✠ **C.H. BENNETT**, DAMIEN QUERLIOZ and JACQUES-OLIVIER KLEIN, "Energy, Speed, and Accuracy Tradeoffs in On-Chip Learning Systems Built with Memristive Nanodevices" , *GDR Bio-Comp Workshop, Fall 2016*, 2016. *Oral presentation.*

Appendix A

Additional details on TBFe devices

"In this appendix, *additional details relating to the fabrication, mechanisms of switching behavior, and scaling of the TBFe physical nanosynapses are presented.*"

A.1 Synthesis and Fabrication of TBFe Devices

A.1.1 Chemical Synthesis

The iron(II) *tris*−bipyridine complex with diazonium functions was synthesized according to the procedure depicted in Supplementary Fig. S2. All reagents and chemicals were purchased from Aldrich and used as received. 4'-(4-Aminophenyl)-2,2'-bipyridine (**BipyNH$_2$**) was prepared according to a procedure described previously [**?**]. Characterization techniques: NMR spectra were recorded with a Bruker ADVANCE DRX 400 (400 MHz). Chemical shifts δ are expressed in ppm relative to tetra-methylsilane (TMS). Infrared spectroscopy (IR) was realized with a Bruker Vertex 70 spectrometer (resolution 2 cm^{-1}, 24 scans collected, MCT detector) equipped with a Pike Miracle plate for ATR. UV-Vis spectra were recorded with a Perkin Elmer Lambda 650 spectrometer. Mass spectra were acquired in the positive mode on a LCQ-ion trap Thermofinnigan spectrometer equipped with an electrospray source (MS-ESI).

a) [Fe(Bipy−ph−NH$_2$)$_3$][PF$_6^-$]$_2$ (**FeNH$_2$**).

A solution of iron(II) tetrafluoroborate hexahydrate (91 mg, 0.33 eq.) and **BipyNH$_2$** (200 mg, 0.81 mmol) in ethylene glycol (4 mL) was heated at 60°C for 5 min. Afterward, 100 mL of water saturated KPF$_6$ were added. The precipitate obtained was filtrated and washed several times with diethyl ether to give **FeNH$_2$** as a purple solid (270 mg; 92% yield). ^{1}H NMR (400 MHz, DMSO-$_6$, δ): 9.12 (d, J=7.9Hz, 1H), 9.01 (s, 1H), 8.23 (t, J=7.2Hz, 1H), 7.83 (d, J=8.0Hz, 2H), 7.76 (m, 1H), 7.60-7.45 (m, 2H), 7.28 (d, J=6.1Hz, 0.5H), 7.17 (d, J=6.1Hz, 0.5H), 6.69 (d, J=8.0Hz, 2H), 5.87 (s, 2H). ^{13}C NMR (50.32 MHz, DMSO-d$_6$, δ): 159.3, 158.6, 151.7, 149.0, 138.2, 130.6, 128.8, 128.3 (2C), 127.3, 124.0, 122.1, 120.5, 118.8, 113.8 (2C). IR ν = 3367, 3098, 1594, 1524, 1470, 1437, 1411, 1330, 1261, 1189, 1054, 826, 787 cm^{-1}. UV-vis (*acetonitrile*): λ_{max} = 545, 508 (sh), 369 nm. MS (ESI) m/z: calcd for C$_{48}$H$_{39}$FeN$_9^{2+}$, 853.20; found, 398.8 (M - 2PF$_6^-$).

b) [Fe(Bipy−ph−N$_2^+$)$_3$][PF$_6^-$ or BF$_4^-$]$_5$ (**FeN$_2^+$**).

Under argon, nitrosium tetrafluoroborate salt (13 mg, 1.2 eq.) was added directly to a degassed solution at -40°C of **FeNH$_2$** (100 mg, 0.1 mmol) dissolved in dry acetonitrile (5 mL). After 5 min of stirring at this temperature, diethyl ether was added until a precipitate came out. The precipitate was filtrated, washed several times with diethyl ether to give a purple powder **FeN$_2^+$** (125 mg, quantitative yield). ^{1}H NMR (400 MHz, CD$_3$CN, δ): 8.88 (d, J = 4.2Hz, 1H), 8.75 (dd, J = 7.5Hz, J = 4.5Hz, 1H), 8.67 (d, J = 8.7Hz, 2H), 8.34 (d, J = 8.7Hz, 2H), 8.21 (t, J = 7.5Hz, 1H), 7.80-7.40 (m, 4H). IR ν = 3107, 2280 (N≡N), 1583, 1540, 1468, 1438, 1402, 1333, 1285, 1233, 1022, 826, 785, 750 cm^{-1}.

A.1.2 Electrografting of Iron complex

The electrochemical grafting was conducted in a single-compartment three-electrode cell with a potentiostat (Model VSP Bio-Logic SAS) in a glovebox. Ag/AgNO$_3$ (10 mM) electrode and a platinum wire served as reference and counter electrode, respectively. All potentials in the fol-

lowing are referenced to $Ag/AgNO_3$. The silicon substrate with the patterned gold working electrodes was completely immersed in a solution of $\mathbf{FeN_2}^+$ (34 mg/L) dissolved in tetrabutylammonium hexafluorophosphate (0.1M)/acetonitrile electrolyte. The gold electrodes were connected with a passivated tungsten tip. Chrono-potentiometry technique (5s at -8μA) was used to make smooth thin films of covalently bounded Iron(tris-bipyridine) (**TBPFe**) complexes (See Supplementary Fig. S3), AFM image and height profiles of modified electrodes. Cyclic voltametry (CV) technique was used to grow thicker film for the memristive device.

The electrochemical properties of an electrodeposited 14 nm-thick film were studied by CV in a medium of analysis free from the pristine complex (Supplementary Fig. S4). CV shows the characteristic peaks of the complexes grafted on the electrode, i.e. a reversible wave in oxidation (0.78 V vs Ag/Ag^+), two reversible waves in reduction (- 1.58 V and -1.76 V). It is worth to note that the potentials of the metal complexes inside the polymer films were closed to those for dissolved iron trisbipyridines complexes.

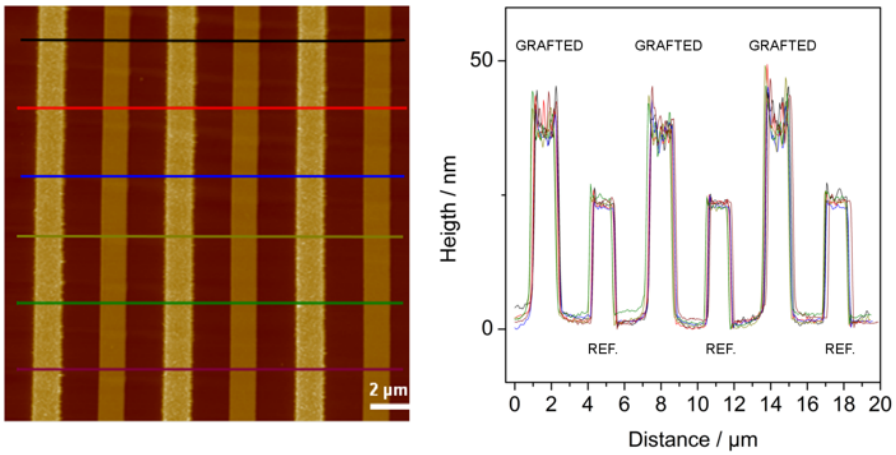


Figure A.1: AFM image of electrodes electro-functionalized with $\mathbf{FeN_2}^+$ (2×10^{-5} mol·L^{-1}) in 0.1M NBu_4PF_6/acetonitrile electrolyte using chrono-potentiometry technique (5s at -8μA). Electrodes labeled *REF.* were not grafted and serve as reference. The AFM height profiles show the homogeneity of the films thickness along each electrode and between electrodes grafted separately.

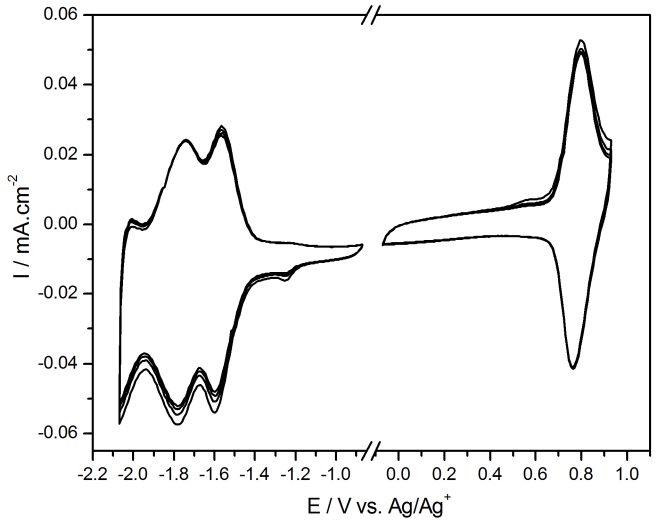


Figure A.2: CV of a gold electrode modified by $\mathbf{FeN_2}^+$ in 0.1 M Bu_4NPF_6/acetonitrile, 100 $mV{\cdot}s^{-1}$.

A.2 Device endurance and stability

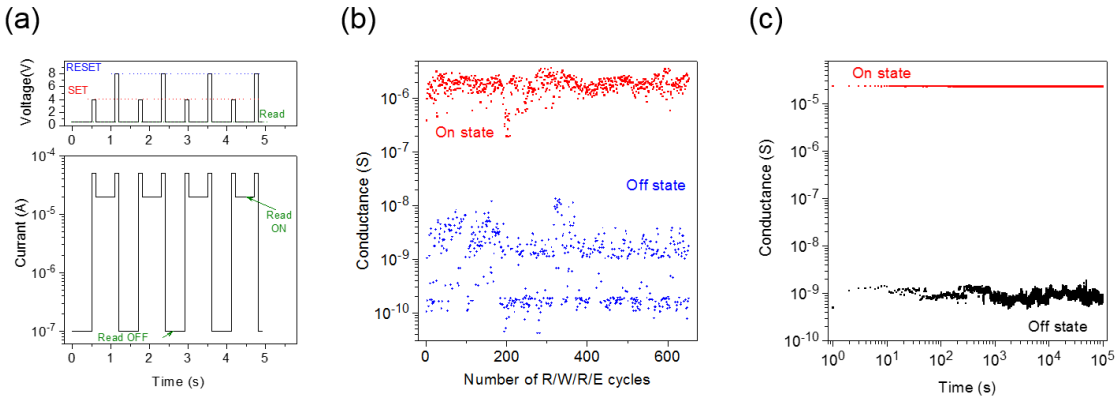


Figure A.3: (a) Demonstrates current and voltage scheme used to conduct, SET, RESET, and Read operations (b) Demonstrates the ability of devices to withstand several hundred progressive pulse programming test schemes (Read/Write/Read/Erase) (c) Demonstrates the temporal stability of devices' conductance G over time, once placed in the ON/ G_{Max}, OFF/ G_{Min}, state.

A.3 Crossbar-compatible scaling (Vertical device structures)

The vertical memristive junctions are fabricated by electrografting a thin organic film (of typically 20-30 nm) on bottom gold electrodes and then by direct deposition of a second gold electrode. Both bottom and top electrodes are fabricated by e-beam lithography, evaporation and

lift-off. A schematic representation and a SEM image of such vertical device are shown in figure A.4a,b. After electrical forming, the vertical device exhibit comparable characteristics to the planar ones (figure A.4c): similar SET (2.5-3V) and RESET (3.7-4V) bias ranges and a wide range of accessible intermediate conductivity states. The endurance of such vertical configuration is lower than for planar structures and the device failure is characterized by devices getting shorten. This is most probably due to partial deterioration of the organic layer upon metal evaporation. Work is underway on the electrografting parameters to improve the compactness of the film to limit such degradation.

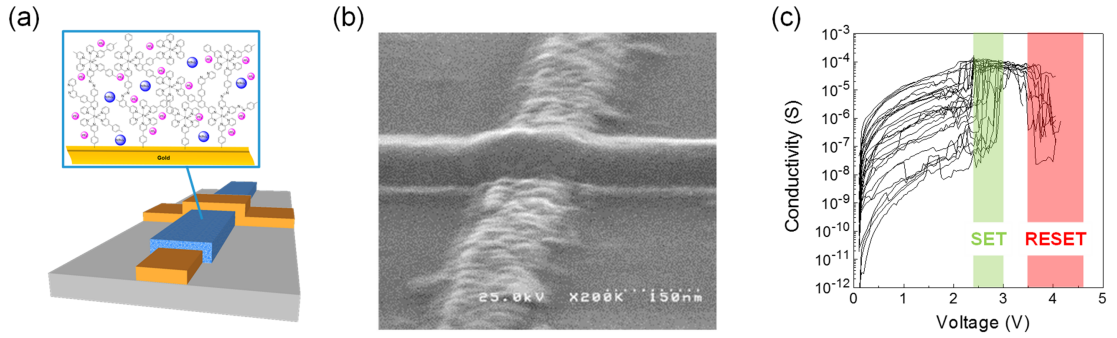


Figure A.4: (a) Schematic representation and (b) SEM image of a vertical memristor. (c) Representative IV characteristics of the vertical devices upon application of a series of voltage sweeps with different final states. The conductivity at a read bias of 1V shows the accessibility of different intermediate states.

A.3.1 Causes of Filamentary Operation

There is little agreement in the literature on the filamentary vs. bulk nature of the switching and on the respective impact of the different elements (electrodes, organics, and substrates). In our case, the unipolar filamentary nature of the switching is clearly established based notably on the two following arguments:

- Firstly, the device properties (threshold voltages and the maximum conductivity) do not scale with the junction area. As displayed in figure A.5, devices with different junction area spanning a very large range display very similar characteristics. This implies that the change of conductivity only affects a small area rather than the entire junction.
- Secondly, the RESET threshold varies with the initial conductivity, while the SET threshold does not. The higher the initial conductivity, the higher the voltage required to RESET the device. This can be explained if the organic memristor varies its conductivity through the formation and rupture of conductive filaments. It is indeed expected that when a larger conductive filament is formed, a higher energy is needed to break it.

Concerning the nature of the filaments, we excluded the role of the metal electrodes by changing the nature of one or both metals (including by using carbon nanotubes as electrodes).

We also excluded the role of the surface (in planar junctions) notably by studying organic memristors on flexible organic substrates. Electrochemical characterizations show clear reversible memory effects in solution associated with redox process. Yet, in a device configuration, we cannot presently fully exclude the formation of carbon-based filaments originating from a degradation of the redox film during the electrical forming step.

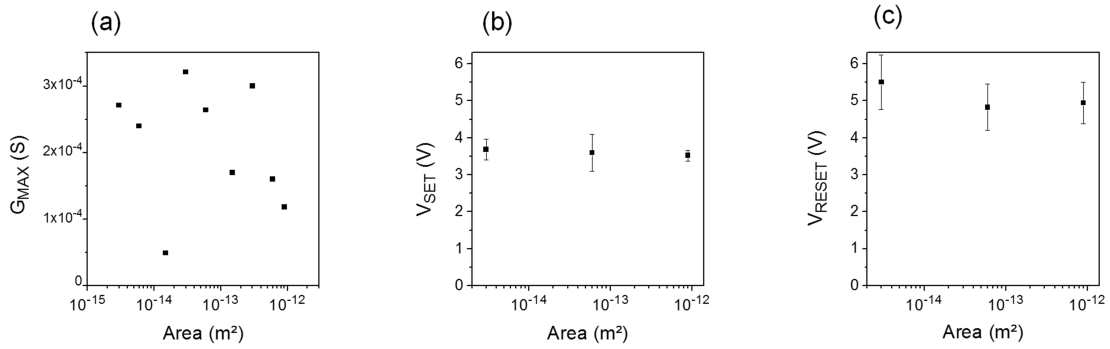


Figure A.5: Dependency of (a) the maximum conductivity (G_{MAX}), (b) SET threshold and (c) RESET threshold to the cross section area of the channel.

Appendix B

Additional learning experiences with TBFe devices

"THIS *appendix introduces in more depth the set-up used to implement hardware learning with the organic nanodevices. Moreover, it contains additional companion simulations to the experimental works not all visualized within the main text.* "

B.1 Additional examples of SR/SO Learning

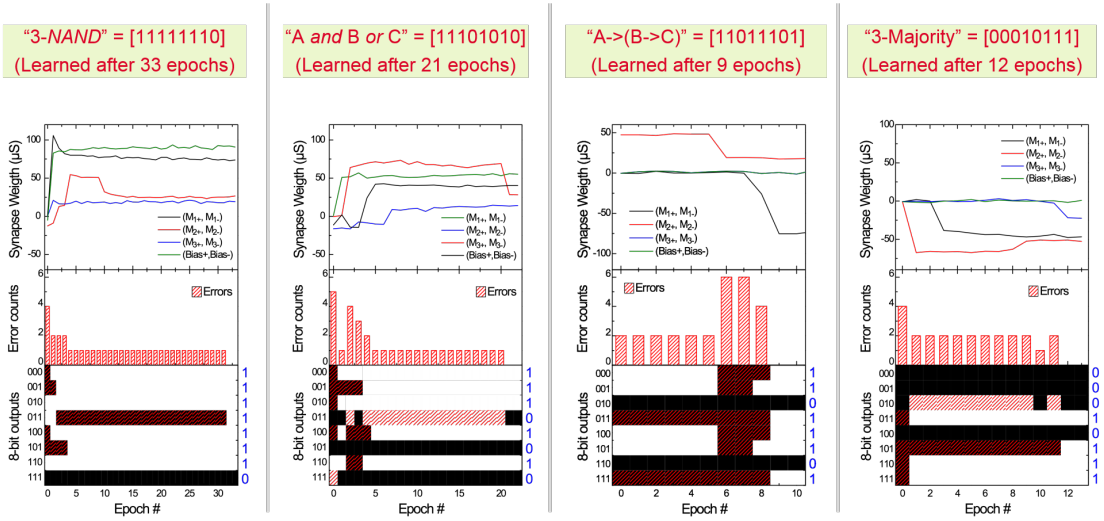


Figure B.1: Examples of learning 3-input logic function using SO Mode. For each example, the top panel shows the evolution of synaptic weight; the middle panel shows the error counts after each epoch; the bottm panel shows the digital output of each case after each epoch, where error(s) are marked in red. As shown in the middle panels, the system often get stuck at an intermediate state with one or two errors due to the "nonlinear SET" behavior of the memristor.

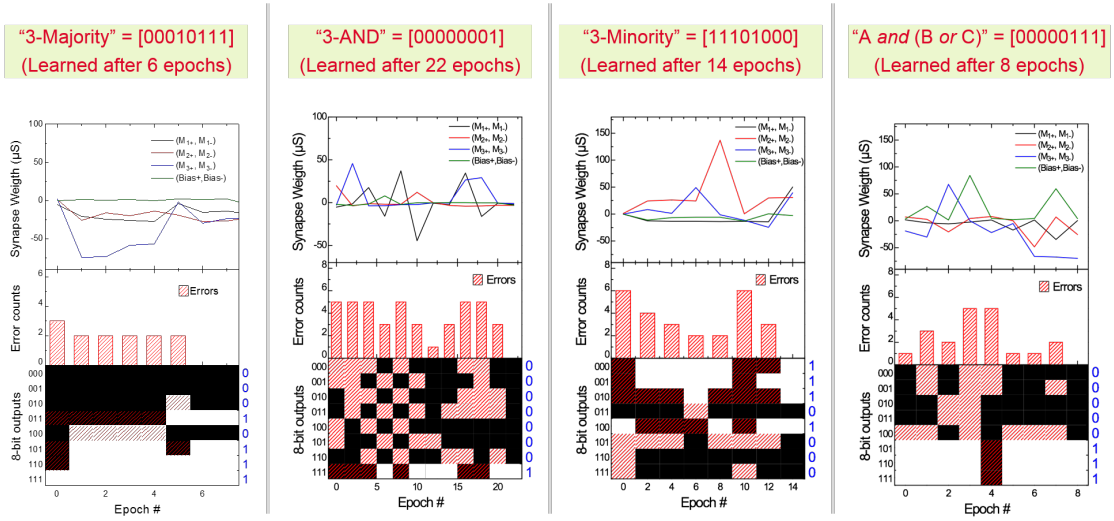


Figure B.2: Examples of learning 3-input logic function using SR Mode. For each example, the top panel shows the evolution of synaptic weight; the middle panel shows the error counts after each epoch; the bottm panel shows the digital output of each case after each epoch, where error(s) are marked in red. As shown in the top panels, there is more fluctuation of the synaptic weights when using Mode 2.

B.2 Complementary learning simulations

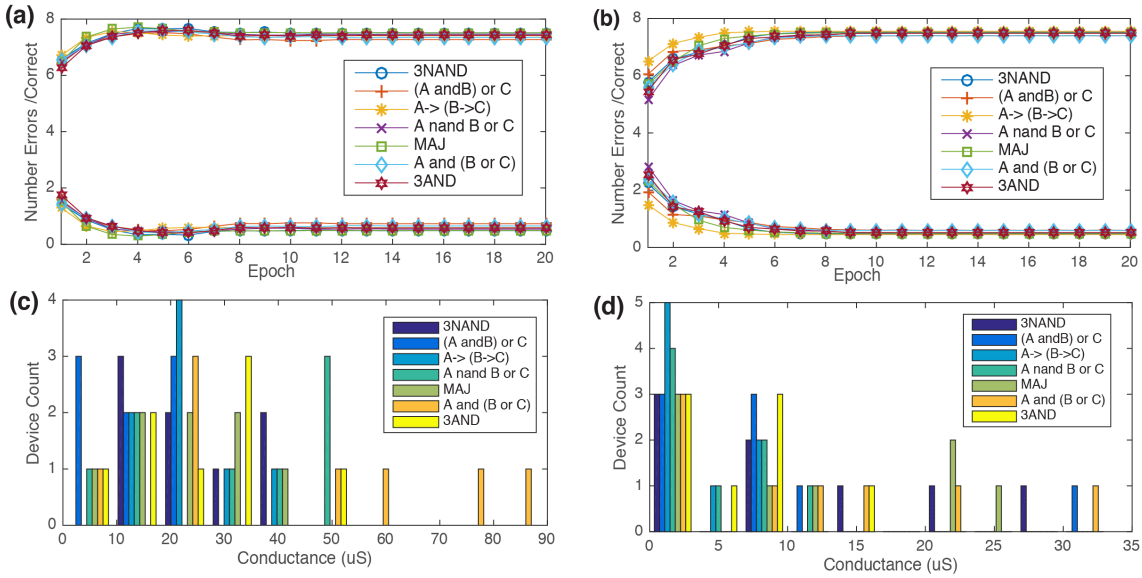


Figure B.3: Simulated learning cases using SO (a) and SR (b) learning with variable nanodevices ($\sigma(G_{\text{Max}}) = 40\%$, $\sigma(V_{t1}) = \sigma(V_{t2}) = 10\%$). Learning is imperfect and success varies slightly based on the function. Concluding conductances for a characteristic single iteration (c) are noticeably higher than those for a characteristic SR iteration (d). $\Delta G_+, \Delta G_-$ vary on a device by device basis but always pegged at $10\% G_{\text{Max}}$ for the given device.

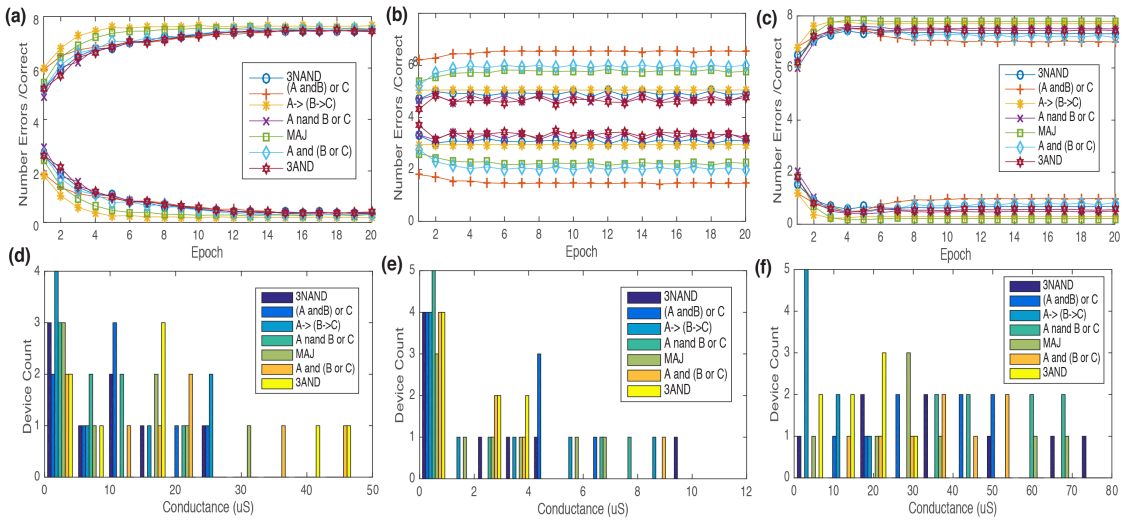


Figure B.4: (a) Minor SR asymmetry case ($\Delta G_+ = 15\%G_{\text{Max}}$ $\Delta G_- = 20\%G_{\text{Max}}$) closest to experimental case and listed in Table 2 paper, and characteristic weights (d); (b) SR performance is severely impacted when $\Delta G_+ = 5\%G_{\text{Max}}$ $\Delta G_- = 20\%G_{\text{Max}}$, as concluding weights (e) are uniformly low; (c) Performance is far superior with inverse asymmetry ($\Delta G_+ = 20\%G_{\text{Max}}$ $\Delta G_- = 5\%G_{\text{Max}}$), and weights much higher (f). Device variability same as Fig. S7 in all cases.

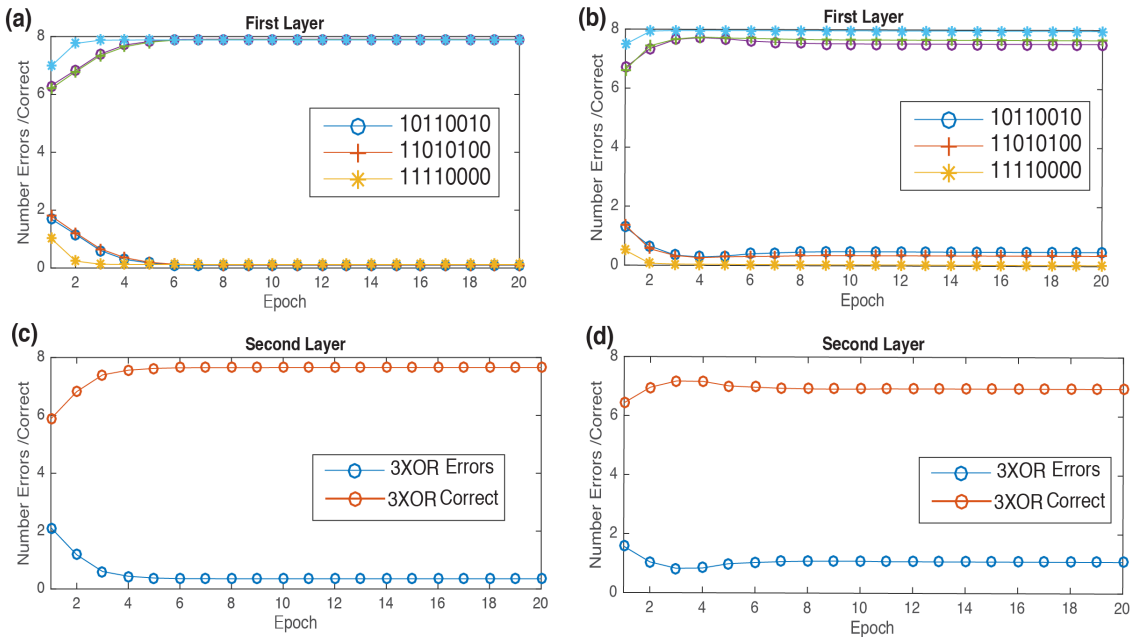


Figure B.5: SR mode learning of a non-linearly separable function- the 3XOR function- is depicted in (a) for the composing functions and (c) for the final function. Curves represent mean errors/correct values during each epoch for 500 monte carlo simulations; learning of the first layer functions is nearly perfect, while the second layer function (3XOR) learns in 79.6% cases by mean 30.2 epochs. SO mode learning for the identical task is depicted in (b), (d) for first and second layers respectively. 71% of all cases now learn successfully, at a faster mean 23 epochs. Every simulation assumed device variability at the same levels examined in Figs. S10-S11; $\Delta G_{+,-} = 10\% G_{\mathrm{Max}}$ and the SR case is symmetric at this level.

Appendix C

Online memristive anomaly detection scheme

"An additional application of evolving memristive tasks to a different task besides generic classification is described, simulated, and evaluated in this section."

C.1 Background: methods to implement novelty detection

Anomaly, or novelty detection- determining whether a given input is substantially different then previously seen samples- is a problem with abundant applications in the fields of visual and other sensory data processing (robotics), complex time series analysis (financial and industrial analysis), and monitoring of complex systems (e.g. web traffic or server bandwidth). The problem is statistically represented as a classification task involving an outlier; while a wide variety of generic mathematical techniques such as extreme value theorem (EVT) exist to solve the problem with stable training sets, online (real-time) solutions to the problem are difficult [313]. While online machine learning algorithms do exist to tackle the problem, the computational requirements are steep since each recursive computations are required at every timestep; moreover, the system has difficulty being correctly tuned to the problem [314]. Local circuits implementing anomaly detection are especially attractive for Internet of Things (IoT) devices and space and ocean exploration vehicles, yet existing designs implementing these functions are space and power hungry [315, 316]. In contrast, the brain- in particular structures such as the left visual field of the right hemisphere- continuously provides robust online detection in response to stimuli such as alarming or impossible objects [317, 318].

Cognitive computing could close this massive performance gap through a combination of superior computing materials, such as synaptic nanodevices, as well as more approximate algorithmic approaches [319, 320]. In real distributed systems, signal corruption or sparsity, high dimensional sensor input/task, and strong energy limitations make the online anomaly detection task intrinsically difficult. Kohonen and Oja's novelty filter- a system that continuously extracts the orthogonality of new data (input vectors) to the previously stored patterns that have been stored in a memory matrix[321, 322]- nevertheless manages to tackle the task by computing in memory. In this work, we show that a crossbar of memristive nanodevices can implement the key principles of this filter when inputs are presented recurrently or coincidently. Recurrent memristive crossbars have been examined from a mathematical perspective [323]; within a neuromorphic context, such a system has only been preliminarily analysed [324]. Directly implementing a novelty filter with a crossbar of memristive nanodevices, to our knowledge, has not been proposed before.

C.1.1 Autocorrelation Matrix Memory

Kohonen's autocorrelation matrix memory is an analog auto-associative system whose key component is a matrix of adapatable weights (M), and whose output $\boldsymbol{y}$ in response to an input vector $\boldsymbol{x}$ at any time t is a combination of the immediate value of that signal joined with the feedback of all past signals, expressed through the weights ($\boldsymbol{y} = M\boldsymbol{x}$). Each weight at row i, column j ($M_{i,j}$) ought to evolve according to the unique influence of its input row (x_i), recurrent or co-

incident column ($\bar{x}_j$), and implicitly past events (starting weight). When $M_{i,j}$ only changes in response to coincidence of active input on x_i, $\bar{x}_j$, the system stores an auto-correlation matrix and constitutes a dynamic novelty filter (can continuously report similarity of the presented input to the auto-correlation). An ACMM may be complete- storing n^2 values relative to an input with n channels, or incomplete- storing only n.

C.2 Online anomaly detection nanoarchitecture

Initially, address-Event Representation (AER) circuitry converts real-time sensor data into frames of voltages spikes $\boldsymbol{x}$. A transposed voltage spike frame $\boldsymbol{x}^T$ may be presented either recurrently, traveling across top wires before flowing to charging circuits that generate appropriate spikes on bottom nanowires, or coincidentally, bypassing the crossbar. While the coincident scheme is sufficient to implement an ACMM, the ability to naturally implement a characteristic delay function at the 'hidden layer' is then lost. Memristive nanodevices at each crosspoint (intersection of a top and bottom nanowire) form filaments in response to sufficient biases, and this internal state variable, readable as a characteristic conductance (G), bestows intrinsic memory of past states (top right, Fig. C.1). In the crossbar architecture, voltage difference across the top and bottom nanowire is exploited to realize a conditional programming scheme. As mentioned, an autocorrelation matrix results when only those devices with both an active input (positive) and active feedback (negative) may non-linearly increase their conductances, G (circled cross points, Fig. C.1). The voltage differential across only these devices should be greater than a characteristic device threshold V_{th}; one scheme that implements this is shown in Fig. C.1, bottom right inset.

In the recurrent case, initial input from top wires is fed to intermediate circuitry as current (for instance, Mead's axon-hillock circuit[32], Fig. C.1 middle left). Since a negative voltage spike is generated after a characteristic integration time (t_{pre}) given by the capacitor, devices evolve only within a brief window t_g (when input voltage spike and recurrent spike are coincident). During t_{post} only bottom nanowires are biased, and during t_{c} capactive circuits reset and no spikes are applied. After each update a subsequent read period (t_r) provides an online estimation of similarity for the previous example. Output currents ($\boldsymbol{y}$) are automatically obtained by the dot product between input voltage spikes at V_{read} and nanodevice conductances. Subsequently, trans-impedance amplifiers convert output currents into voltage at each output, whose values are compared to characteristic thresholds in leaky integrate and fire (LIF) CMOS neurons at each output (Fig. C.1(a)). These neurons ought to feature a characteristic threshold and a short refractory period. In addition to LIF, log-domain LPF as well as DPI CMOS neurons may implement these functions [30]. Our nanodevice memristive model is a generic bipolar device. Our time-step simunlator tracks whether potential on each device during each time integration step exceeds the critical threshold $V_{th} = 0.5V$; if it does, conductance increases non-linearly (ΔG) towards a maximal conductance level ($G_{max} = 10mS$). Active input spikes

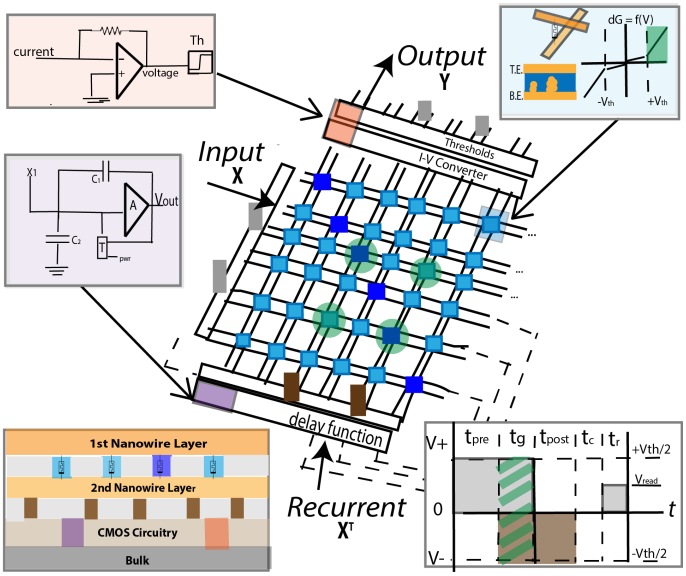


Figure C.1: A complete ACMM system that implements novelty filtering is depicted in the center; diagonal (cobalt) devices correspond to the incomplete ACMM. Orange inset (top left) highlights trans-impedance and threshold circuitry; purple inset (middle left) depicts capactive charging circuit for generating recurrent column spikes; blue inset (top right) shows nanodevice conductance evolution as a function of voltage and filamentary formation. Bottom panels depict a conceptual nanofabric realization of the system (left) and waveforms showing the timing and levels of various spikes used to implement the scheme (right). At this moment, characteristic input (gray) and recurrent (brown) spikes grow only the highlighted (green) device's filaments.

and inverted recurrent spikes are set at $+V_i = 0.25V$,$-V_j = 0.25V$ to fulfill the scheme, while $V_{\text{read}} = 0.1V$. A time step simulator software program tracks the evolution of all inputs, device conductances, and outputs.

C.3 Online anomaly detection demonstration

C.3.1 Evaluating Filter Performance

Images of 36 pixels were subsequently presented to the system in spike form in three epochs, each consisting of 40 cycles as depicted in Fig. C.2. An 'X' (class 1), 'O' (class 2), and 'Z' (class 3) were presented in that order to the network. All images have the same number of active pixels (8), yet they are dis-similar to each other to varying degrees. While classes 1,3, and 2,3 each have a Hamming Distance of 8 (pixel flips relative to each other), classes 1,2 have a distance of 16 (are completely orthogonal to each other). Significantly, class 3 can be built entirely from the pixels of classes 1,2. During online learning, memristive devices evolve and reads report disparities through time and within the input space relative to what is stored in auto-correlation. If the

novelty filter operates correctly, we expect to see sharp drops in reported similarity- or high orthogonality- during transition periods (when a new image is shown). Over time, the ACMM will eventually learn the new image as well and return to high similarity. A false positive error is defined as reporting an anomaly when none existed; a false negative is missing a true anomaly.

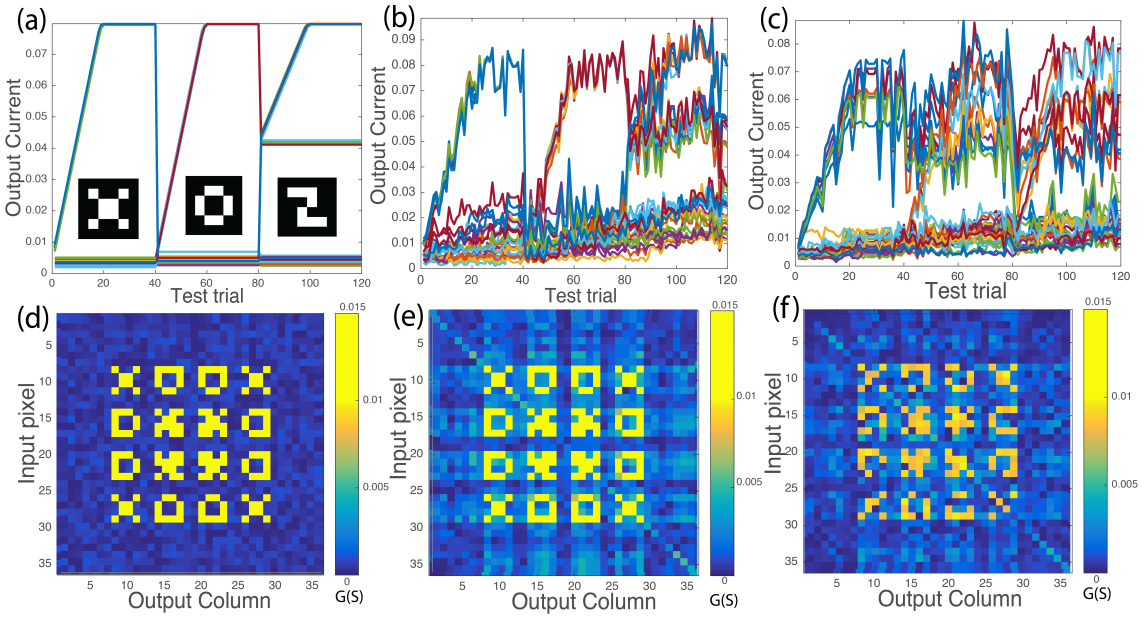


Figure C.2: Top row: output currents at all columns ($\boldsymbol{y}$) is depicted for the perfect case (a), imperfect case with only 5% noise (b), and imperfect case with 5% noise as well as device dispersion. Bottom row: corresponding conductance heat maps showing stored conductances for all n^2 at the last moment of the network's operation ($t = 120$).

C.3.1.1 Complete Adaptive Matrix Memory

In a complete ACMM, the response of every output neuron corresponds to a unique sub-space of the auto-correlation matrix (M). The effect of this is visible in Fig. C.2(a), where each of the n lines corresponds to the read current of a different column or channel of devices (before voltage conversion or LIF neuron) during 120 test trials. As visible in Fig. C.2(b), a 'fractal' auto-correlation map is stored in the conductances of nanodevices (pictured at the end of testing). While class 1 and 2 appear visually, since class 3 is a subset of both it is implicitly contained. This map produces distinct spiking behavior during all three of the epochs; using the information of all n output channels, anomaly detection is then successfully detected during each transition. The distance that current falls (corresponding to the drop in reported similarity) is half in the case of 2 to 3 versus 1 to 2; this exactly corresponds to the relative differences in hamming distance between the classes. In order to simulate imperfect effects, the following were added to the previously mentioned simulations: for noise, 5% of all incoming pixel pulses sent to the network were flipped (actual pixels were deleted, and non-pixels became pulses);

for variability, 5% variance around earlier mentioned mean threshold, maximum conductance, and characteristic conductance change values were assigned on a device-by-device basis at a normal (Gaussian) distribution. When adding just noise (5%) to the complete system, the primary effect is up and down current oscillations (predicted similarity) within epochs due to random 'up and 'down' pixels; however, neither the overall stored conductance pattern nor the current trajectories through time are substantively changed (Fig. C.2(b),(e)). However, when both noise and device variability are added, the system begins to deviate substantially from the original pattern. The clustering of output neuron current levels to a few quanta, as visible in Fig. C.2(a),(b), is no longer observable in (c). This causes several false positives (an anomaly is reported when the image was the same). Additionally, the fractal map (auto-correlation) of digits being stored in the weights now has random holes in it (Fig. C.2(f)); this follows from the fact that deviations around V_{th} may cause some devices with uncharacteristically high thresholds to never satisfy the condition for conductance increase (e.g., if $V_{th} = 0.6$). While this can be solved simply (increasing V_i), other effects of device variability are more intractable. Since many devices saturate at ON , the effect of $G_{\max}$ variability can be substantially negative to accurate online similarity reporting at a higher dispersion value.

C.3.1.2 Incomplete Adaptive matrix memory

An incomplete ACMM requires only n devices, possibly saving space and energy. To verify successful operation, only diagonal (cobalt devices, Fig. C.1) were also demonstrated on the previous task. Figure C.3 shows operation in incomplete mode for both the perfect inputs and devices case ((a), (c)) , and imperfect devices and noisy inputs ((b), (d)). In the perfect case, high orthogonality (anomaly) is detected at the both the beginning (configuration period), and during the transition from 1 to 2. However, a false negative occurs between epoch 2 and 3; since the auto-correlation matrix has stored every pixel previously in class 3 ('Z') already and is only using $n = 36$ devices, it simply does not have the computational capacity to report this transition. This implies that even with perfect devices and input, an incomplete ACMM only works with relatively dissimilar inputs. In the imperfect case, the combination of noise and device variability creates deviation between output currents within epochs, and leads to two small anomaly detection periods slightly after the transitions from 1 to 2 and 2 to 3. While noise tipped the scale towards successful detection in this case (avoided a 'false negative'), it does not reliably do so; at even higher noise levels, the chance of a false positive grows as well.

C.3.2 Classification Performance

All performances so far used analog current outputs to demonstrate the performance of the filter, however this mode of operation may be difficult in real nanoelectronic systems where analog values are difficult to preserve and compare through time. In a realistic online operation mode, the goal is to read-out a binary signal from the system about whether it is detecting

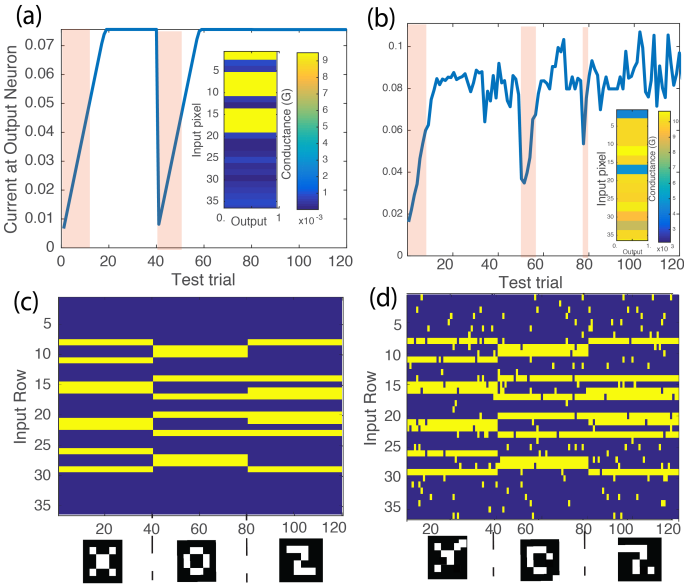


Figure C.3: Top row: output current (y) received when presenting perfect images to perfect network (a) and imperfect images to variable network (b). Zones of anomaly detection, where the LIF neuron would likely fire, are denoted in the shaded regions, and the small bar charts hold a record of the final conductance of each of the n nanodevices. Bottom row: input spikes presented to generate weight changes over all trials for perfect (c) and imperfect (d) trial.

an anomaly situation or not at any particular state t; *[e.g.]* whether the observed state is the baseline/normal state (0) or anomaly state (1).

C.3.2.1 Implementation of a binary read-out for novetly detection systems

The online binary mode is demonstrated in Fig. C.4 (a) which shows that such a scheme is only effective at determining moments of change from a stored baseline; it misses to continuously report error during the 'Z' epcoh. Worse, such an analog (baseline) value would need to be stored, which would necessitate more circuit overhead. However, a slight modification can yield more accurate binary estimates of novelty detection. This can occur if the currents are summed, and a single value t_{ref} is stored and compared against the output of each line y_i. Electrically, this can be implemented through a simple circuit, such as a comparator clamped to the target value, and as needed, a following neuron (e.g., Leaky Integrate and fire design). As visible in Fig. C.4(b), this system now performs better, missing only a small moment of anomaly case in the 'X' epoch. A final option requires that the vector of outputs are compared to a vector of baselines $\boldsymbol{t}$, so that the system now compares against n threshold instead of one. As visible in Fig. C.4(c), the system is now able to always report anomaly outside of its baseline state. Note that, for both the two readout/threshold systems, a configuration step is required before the system can enter into 'detection' mode (threshold must be set).

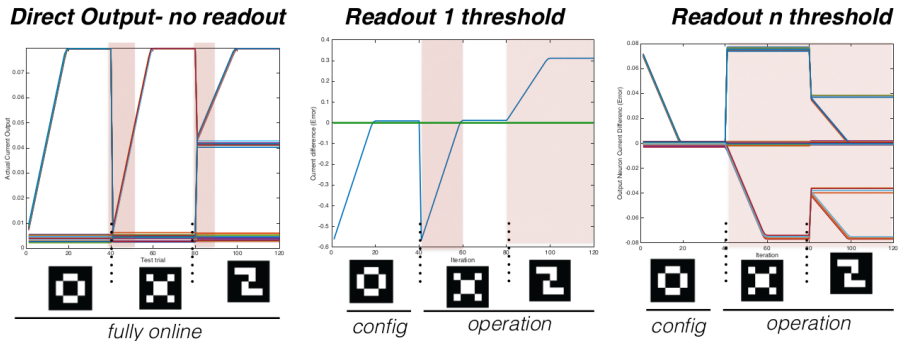


Figure C.4: Left: Direct current output is used to evaluate anomaly cases. As visible, only the transient moments are noted as anomalies (red). Middle: A sum of outputs is compared to a single threshold. As visible, most of the anomaly period is detected. Right: each output channel has a threshold. Now, the entire anomaly period is noted.

C.3.3 Novelty Detection with Forgetting

In Kohonen's original work, it is already mentioned that the standard novelty detector engine suffers from a critical draw-back, which is over-saturation due to pattern imprint in the memory matrix [321]. If synapses can naturally relax or forget, the system's capacticity is massively improved through time/operation (note however that its memory capacity at any one moment, however, is not greater). Finally, we tested enhancing the standard novelty filter to a novelty filter with the forgetting effect by integrating the ECM device with meta-plastic behavior- earlier introduced in Section 4.2- into the already described system. Figure C.5 shows, on the left, the relaxation of all $n = 64$ synapses in the considered memory matrix back towards their resting state following a previously occuring memory imprinting and detection phase (what is demonstrated in Fig. C.4). The results suggest that a refresh time of about $0.1s$ is needed by the system to partially erase weights, and $0.5s$ to completely erase/relax. This is a long time from the device perspective, but possibly an acceptable refresh window from the nanosystem operation point of view. Next, we tested whether the forgetting or relaxing effect has a deleterious effect during operation itself. As visible in Fig. C.5 left, despite very small momentary relaxations, the direct current outputs remain correct and so integration with the designed system is indeed possible.

Lastly, we tested integrating a full read-out scheme for anomaly detection as earlier discussed with the fully non-volatile device. For both the one threshold (Fig. C.6, left) and multi-threshold schemes (Fig. C.6, right), anomaly is detected continuously during the two periods corresponding to the non-baseline shown image. Note in these diagrams the combined threshold , and one of the many, are shown as green line(s).

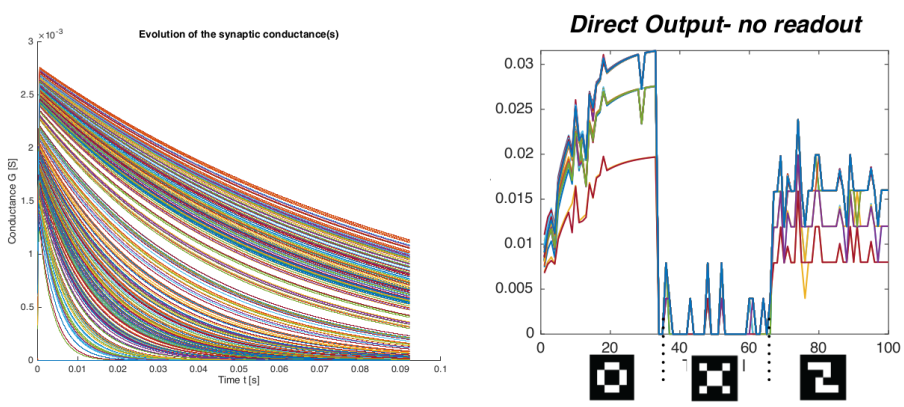


Figure C.5: Left: forgetting or relaxation of nanosynapses is demonstrated after the main anomaly detection phase. Right: basic operation of the novelty filter is confirmed with the ECM cells. As visible, current output drops to none during the second image class where no pixels are shared; in contrast, during the last phase, the distance is estimated as half (half of the pixels are stored in the same image).

C.3.4 Discussion and Future Work

The proposed systems successfully implement online novelty filtering and generally demonstrate a trade off between circuit overhead and usefulness; in particular, analog current output reveals distance between inputs, but this distance needs to be stored and/or evaluated with off-crossbar CMOS or logic circuitry in order to be meaningfully used in the novelty detection set-up. In addition, the chosen image anomaly task was extremely simple and intended as a proof of concept. Real AER data, such as from a Dynamic Vision Sensor (DVS) camera, may prove a more challenging task.

Robustness of these systems to imperfections is mixed; nanodevice variability (especially around G_{Max}) is more damaging than noise. At the modest dispersion levels simulated here false positives were possible (Fig. 2(c)); at even higher levels, similarity measures may become meaningless. Thus, reliable nanodevices with low dispersion should be preferred for physical realizations of the system. As all devices in the present scheme will tend to saturate towards G_{Max} after many patterns have been presented, a RESET could be periodically applied on all input and output wires after awhile (*e.g.*, erase the entire ACCM). However, if memory traces are treated as valuable by the system, the novel detection with forgetting scheme (Section C.3.3) could significantly extend the lifetime of such an on-chip novelty detector. In particular, the use of a device with both volatile and non-volatile filamentary dynamics allows for natural relaxation (forgetting)[293] beyond a certain memorization moment intrinsic to the device physics. This scheme certainly deserves further exploration, particularly with respect to its potential resilience to device variability and how it may be used in other contexts.

Finally, we note that, as formulated, the present simultaneous voltage encoding on the crossbar implements a massive logic function - an AND gate at every cross-point. Recently, an

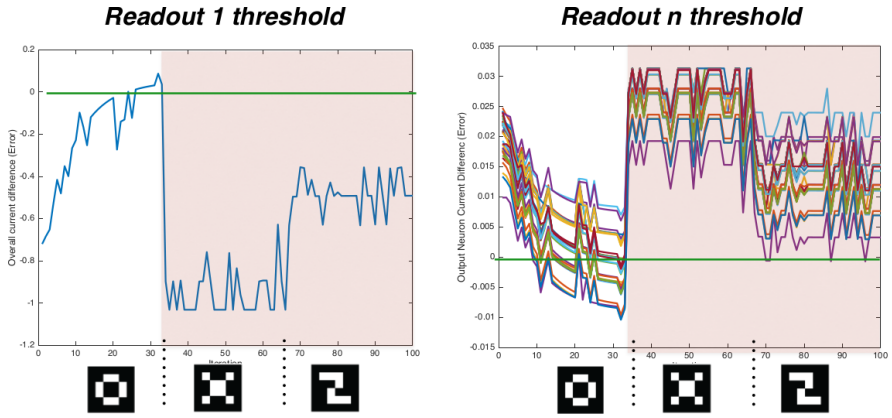


Figure C.6: The single threshold (Left) and the Multiple threshold (Right) binary anomaly filters is demonstrated during operation phase with the ECM device. The green line represents a single threshold.

extension of this work demonstrated that unipolar devices with an appropriate voltage scheme could also implement the XOR gate at the diagonal elements of the crossbar (cobalt devices, Fig. C.1)) [325]. Further extension of the present work could be to explore how different logic gates could be multiplexed through time and space to implement other functions besides anomaly detection.

Appendix D

Synthèse en Français

"ICI *nous présentons brièvement contexte, et résumons rapidement les résultats des travaux expérimentaux et théoriques de la thèse.* "

Introduction et contexte

À l'ère de l'informatique distribuée, les algorithmes cognitifs inspirés par le cerveau effectuant des opérations d'inférence localement plutôt que sur des serveurs déportés présentent de nombreux avantages. Ces avantages découlent notamment de la consommation électrique des serveurs informatiques centraux, et son impact environnemental indirect, ainsi que des problèmes de sécurité et de fiabilité liés à la vulnérabilité des systèmes centralisés quant aux attaques et perturbations. Cependant, même si les technologies informatiques existantes, par exemple les smartphones, les tablettes et nouvelles variétés d'appareils et de capteurs «connectés» effectuaient des calculs localement, de fortes dépenses énergétiques sont attendues pour répondre à l'explosion de la demande de l'Internet des objets. Ces constats résultent à la fois d'inefficacités physiques dans les dispositifs à transistors modernes et d'une faiblesse conceptuelle fondamentale dans la structure actuelle des ordinateurs modernes dans lesquels la mémoire est physiquement séparée des opérations de calcul - le «goulot d'étranglement de von-Neumann». Dans ce contexte, la convergence de nouveaux équipements matériels économes en énergie et d'architectures informatiques à faible coût est une nécessité urgente pour les ingénieurs et scientifiques.

Dans cette thèse, j'explore en profondeur comment l'intégration d'une nouvelle génération de mémoires émergentes non volatiles, communément appelées «memristors», représente un atout majeur pour cette vision neuromorphique . Les avantages de ces nanodispositifs memristifs sont multiples; ils sont intrinsèquement économes en énergie, et leurs états étant non volatils, ils présentent souvent de très faibles coûts énergétiques pour la lecture et l'écriture, et ils peuvent s'intégrer dans des structures extrêmement denses, ouvrant la voie à des structures de mémoire Téraoctets ou Pétaoctets. En ce qui concerne les réseaux de neurones artificiels (ANN), ces structures ont le plus souvent été réalisées dans des logiciels et exploitées jusqu'à présent sur des ordinateurs ou des serveurs. Au cours des dernières années, des implémentations dans des réseaux logiques programmables (FPGA) ainsi que des puces spécialisées construites à partir de dispositifs CMOS appelés ASIC (Application Specific Interface Chips) ont également été envisagées pour mettre en œuvre cette approche, et très récemment l'utilisation de nanodispositifs est explorée. L'intérêt pour ce domaine est naturel puisque la convergence des ANN et des nanodispositifs émergents offre une opportunité de faire des calculs mathématiques cruciaux pour les réseaux de neurones tels que les produits scalaires et les fonctions non linéaires, au sein de dispositifs et de nano-structures. Ce sont des tâches qui requierent beaucoup de puissance de calcul sur les ordinateurs traditionnels, mais que les réseaux de neurones à nanodispositifs peuvent exécuter naturellement. Cependant, les recherches de pointe sur l'intégration des nanodispositifs dans les structures ANN sont balbutiantes et intègrent rarement le dispositif, le système et les algorithmes.

En utilisant à la fois des travaux théoriques et expérimentaux, cette thèse apporte des résultats importants à une génération émergente de travaux de physique appliquée visant la réalisation physique efficace de réseaux neuronaux artificiels modernes utilisant des dispositifs de mémoire émergents (nanodispositifs memristifs). En particulier, nous avons découvert que pour maximiser les économies d'énergie et l'efficacité de ces systèmes, il fallait maximiser les capacités intrinsèques (analogiques et temporelles) de ces appareils tout en gardant à l'esprit leurs limites physiques. À cette fin, une considération majeure a été la conception ou l'optimisation de nouvelles architectures qui résistent ou même exploitent les limitations des nano-dispositifs (variabilité, non-linéarité, etc.). Pour concevoir ces architectures, nous nous sommes inspirés de la littérature en informatique en apprentissage automatique, tout en restant attentifs aux applications des modèles conceptuels aux systèmes d'apprentissage physiques.

Résultats

Tout d'abord, une nouvelle variété de dispositifs de mémoire redox polymériques a été caractérisée et des modèles compacts (programmation analogique écrite au niveau matériel) ont été développés pour évaluer ses propriétés uniques et ses potentiels neuromorphiques. Le coeur du dispositif est une substance polymère unique qui peut former des filaments conducteurs entre des électrodes métalliques (Fig. D.1(a)). En raison de ce comportement et conformément à la théorie électrique memristive, ces systèmes possèdent une variété d'états stables. Ces états de mémoire peuvent évoluer positivement (augmenter la conductance), évoluer négativement (diminuer la conductance) ou rester stables pendant les impulsions de lecture (mode non volatile) comme le montre Fig. D.1(b). La substance polymère est chimiquement déposée sur de petites jonctions; nos dispositifs expérimentaux utilisaient des jonctions verticales, mais les jonctions horizontales (dans lesquelles on peut construire des structures croisées) sont faciles à construire en utilisant le même dispositif. Nos simulations dans le passé ont suggéré la compatibilité d'un algorithme d'apprentissage clé qui rapproche la descente en gradient, un puissant algorithme d'apprentissage automatique, dans un environnement standard de structures croisées (crossbar). Nous avons testé le potentiel d'intégration de ce dispositif organique unique en l'intégrant dans un petit système d'apprentissage autonome. En utilisant une configuration de circuit conçue sur mesure et une programmation automatique du réseau de memristors à l'aide d'un FPGA, notre dispositif organique a fonctionné avec succès comme une porte logique reconfigurable. Nous avons démontré expérimentalement qu'il pourrait effectivement apprendre et répéter plusieurs portes logiques à la demande. La mise en œuvre électronique physique est montrée pour Fig. D.1(c) l'un des deux styles d'apprentissage considérés (SO, "Set Only"), tandis qu'une expérience d'apprentissage correspondante correspondant réussie apprenant la fonction logique '00001110' est démontrée en Fig. D.1(d)-(f). De

plus, nous avons démontré que les dispositifs pouvaient effectuer une tâche d'apprentissage automatique standard, la base de données des chiffres manuscrits MNIST. Cette simulation suppose que nous disposions d'une plus grande structure croisée connectée - dix fois plus grandes et interconnectées que ce que nous avons expérimenté. Ce travail, publié dans Nature Scientific Reports, a également exploré les principales limitations des nanodispositifs, telles que la variation d'un appareil à l'autre et le comportement asymétrique des dispositifs entre les modes d'évolution de la conductance. Nous avons constaté que les problèmes d'asymétrie constituaient une limite essentielle, tandis que la variation typique des paramètres inter-dispositifs était beaucoup mieux maîtrisée. En raison de cette contrainte, nous avons développé différents modes de programmation pour mieux prendre en compte le problème d'asymétrie.

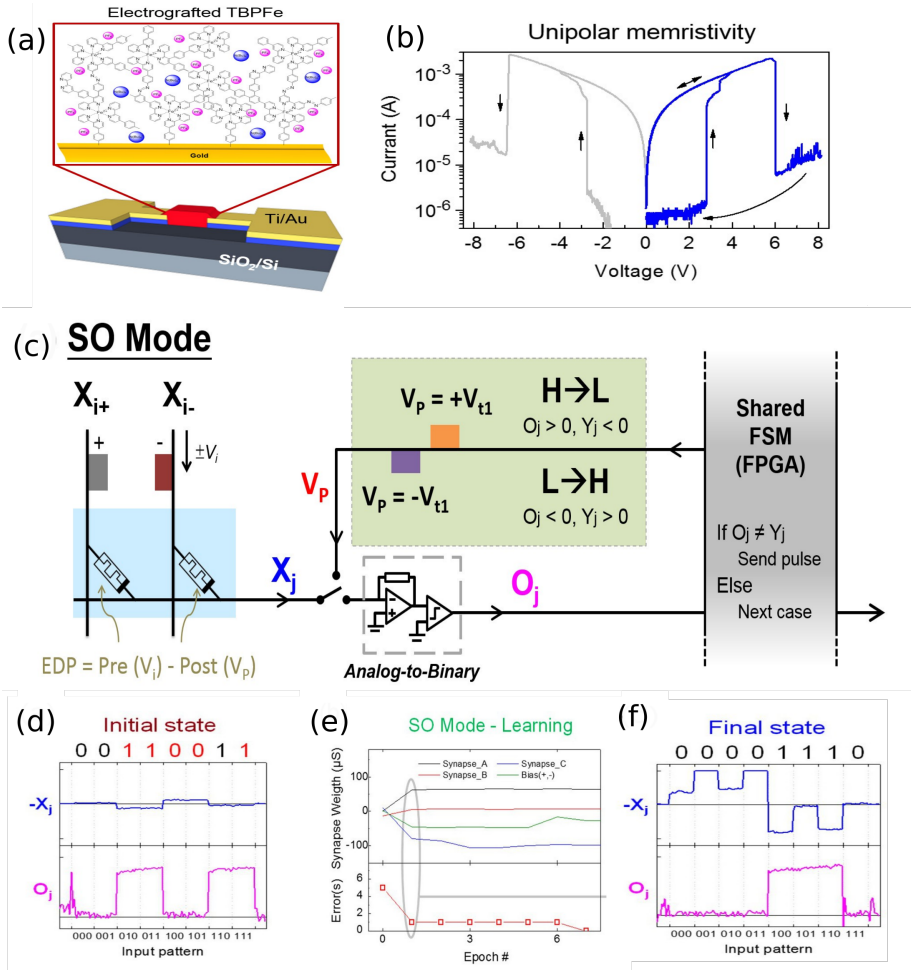


Figure D.1: (a) montre un schéma du nanodispositif organique. (b) montre les caractéristiques électriques de nanodisositif (c) montre l'installation électrique pour l'apprentissage en temps réel (d) montre la sortie avant l'apprentissage et (f) après l'apprentissage est terminé, tandis que (e) montre l'évolution des poids à correctement adapter.

Par la suite, ces résultats ont été élargis dans le cadre d'une exploration plus théorique de systèmes d'apprentissage sur puce plus grands et plus complexes. Comme indiqué dans la littérature, les réseaux monocouches peuvent au mieux résoudre des tâches linéaires et ils peuvent seulement estimer des solutions linéaires pour des tâches plus complexes (tâches non linéaires). Nous avons donc recherché comment deux systèmes classiques à trois couches (une couche cachée) qui peuvent en principe parfaitement résoudre des tâches non linéaires, même très difficiles, doivent être implémentés comme des systèmes d'apprentissage sur puce avec des nanosystèmes memristifs et quelles contraintes critiques ils peuvent rencontrer . Les deux approches que nous avons opposées sont le perceptron multicouche qui apprend par rétropropagation et un réseau d'apprentissage par projection aléatoire, parfois appelé NoProp, également l'ELM (Extreme Learning Machine). Basé sur des contraintes de nanodispositifs réalistes, nous avons proposé des architectures qui permettent à ces systèmes d'apprendre et d'effectuer des inférences en ligne (en temps réel), automatiquement et dans la mesure du possible avec un minimum de circuits annexes. Nous avons également utilisé la même tâche d'apprentissage pour tous les systèmes, encore une fois la base de données MNIST des chiffres manuscrits, et utilisé des modèles paramétrables de dispositifs memristifs qui sont à la fois réalistes physiquement mais généralisables à différentes familles de dispositifs. Lorsqu'ils sont implémentés en systems nanoélectronique, ces structures peuvent apprendre en temps réel en alternant entre des étapes d'inférence ou de feed-forward, des exemples sont passés dans le réseau et un cas d'erreur est obtenu (Fig. D.2(a)), et des étapes d'apprentissage ou de programmation dans lesquelles les cas d'erreur sont améliorés (Fig. D.2(c)).

Pour tous les systèmes d'apprentissage locaux considérés, notre analyse a montré des dépendances critiques sur les contraintes physiques des appareils; notamment, la richesse analogique et la non-linéarité. En ce qui concerne le premier point, la contrainte critique sur la richesse analogique met en évidence la valeur de la construction d'ANNs avec des synapses hautement analogiques plutôt que numériques (binaires) pour faciliter la mise en œuvre de l'algorithme et économiser de l'espace. En ce qui concerne ces derniers, nous avons également constaté que les systèmes ELM davantage bioinspirés possédaient une plus grande résilience aux effets de non-linéarité, tandis que les systèmes d'apprentissage avec backprop étaient plus affectés par ce phénomène. Comme le contraste visuel entre la Fig. D.2(b) et la Fig. D.2(d), l'un des avantages les plus importants du système NoProp est leur apprentissage rapide, qui leur permet d'apprendre à pleine capacité entre 80k-100k mises à jour élémentaires, tandis que les systèmes de rétropropagation totale 500k-1 million de mises à jour à compléter.

Enfin, nous avons examiné comment les conceptions ANN standard peuvent être modifiées pour exploiter les effets temporels, au niveau du dispositif ou du système. La motivation ici est d'utiliser les effets spatio-temporels pour apprendre, tout comme le cerveau. Nous avons examiné trois applications d'apprentissage basées sur le temps et / ou en évolution. Tout d'abord, au niveau de l'appareil, nous avons considéré un autre dispositif memristif,

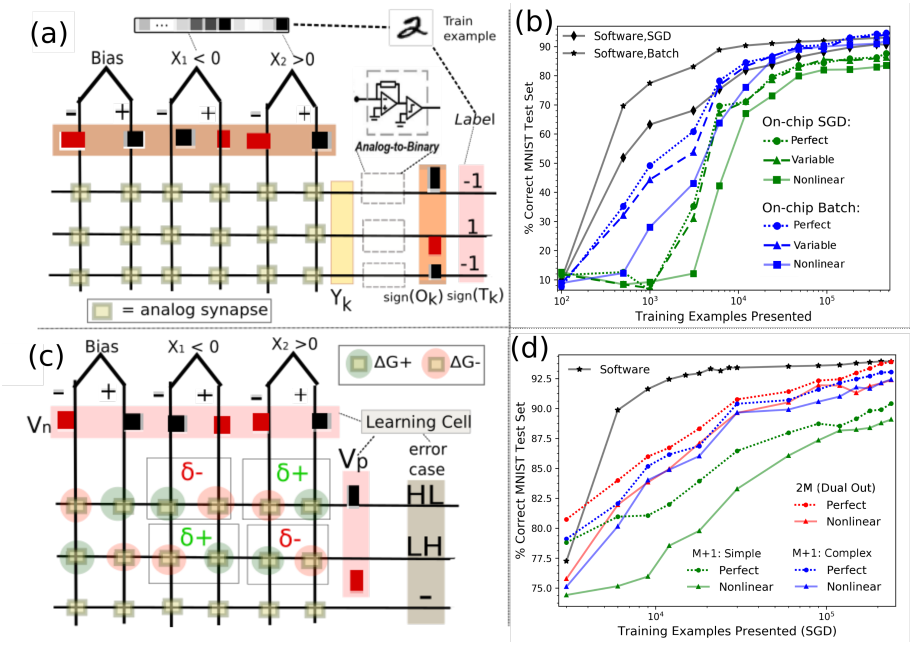


Figure D.2: (a) montre l'implémentation électronique de l'inférence, dans laquelle des situations d'erreur sont obtenues (c) montre l'implémentation électronique de la règle d'apprentissage parallèle ("règle delta") qui les résout. (b) montre l'approche standard multi-couches (toutes les couches entraînées), tandis que (d) montre l'approche NoProp suggérée par l'auteur qui utilise la variabilité naturelle des nanodispositifs et une formation simple aux règle delta.

un dispositif ionique argent qui tombe dans la classe des cellules de mémoire à métallisation électrochimique (cellules ECM). Le travail d'un collaborateur a suggéré que ce dispositif peut émuler une fonction d'oubli bio réaliste et, mieux encore, que le nombre et la variété des impulsions appliquées affectent la vitesse d'oublie. Cette transition entre un mode plastique à court terme dans lequel l'information apprise est rapidement oubliée et un mode plastique à long terme dans lequel elle est conservé pendant des périodes plus longues a été démontrée dans des systèmes d'apprentissage sur une image simple et des tâches de classification d'images complexes, parvenant à être robuste au bruit (activations aléatoires) tout en conservant des reconstructions d'images propres. Cependant, l'appareil dispose d'un paramètre de temps naturel et ces systèmes doivent être orientés autour de ce point, sinon les systèmes risquent la suractivation (activer trop de synapses) ou d'activer trop faiblement les synapses. Il a aussi été démontré que ce système d'apprentissage temporel donnait une amélioration des performances par rapport au système classique de projection à couches cachées (ELM) déjà évoqué.

Deux systèmes d'apprentissage au niveau système exploitant les effets temporels ont également été pris en compte: un système embarqué qui exploite les entrées temporelles et utilise les effets de la couche cachée, apparaissant comme un hybride de l'approche ELM et d'une approximation des réseaux récurrents, connus sous le nom d'ordinateurs réservoir, et un système

de détection d'anomalies qui utilise l'évolution des synapses dans un modèle de grille pour obtenir des estimations de distance en temps réel. Le premier système présente des tâches temporelles intrinsèques à une première couche, qui la projette ensuite aléatoirement sur une variété de filtres temporels sur la couche cachée; ceux-ci peuvent être uniformes, ou variables. Ces filtres mettent en œuvre certaines des propriétés essentielles du calcul à réservoirs, comme projeter des signaux d'entrée dans différents sous-espaces (propriété de séparation). La deuxième couche ne doit effectuer qu'une régression linéaire simple sur puce. La combinaison de ces effets permet au système de fonctionner sur une tâche tempporelle classique, le test de Lyon des chiffres parlés, et de très bien classifier MNIST lorsqu'il est encodé dans une compression de 28 fois (présentée uniquement sur 28 canaux). Le deuxième système de détection d'anomalies, transpose des idées de l'œuvre originale de Kohonen dans les années 1980 à l'utilisation nanoélectronique moderne en considérant les concepts de l'informatique à mémoire. L'utilisation d'un seul crossbar est un environnement idéal pour présenter simultanément des versions originales et transposées d'une seule entrée ou d'un flux d'entrées (canaux), ce qui permet au crossbar de stocker une image de la corrélation des entrées présentes et passées. La distance, le bruit et la variation inter-dispositifs ne nuisent pas beaucoup à ses performances. Un concept complet de détection de nouveauté est testé en utilisant des images simples présentées progressivement; selon la configuration exacte, le système peut signaler avec succès les moments de transition entre les nouvelles classes («scènes»), ou lorsqu'il est connecté à un circuit simple, il peut signaler une anomalie. Enfin, une extension du système est envisagée avec des synapses qui oublient (en utilisant le modèle ECM discuté précédemment), ce qui pourrait augmenter sa capacité.

Perspective

En conclusion, notre travail soutient l'affirmation selon laquelle les percées en nanoélectronique, en particulier le travail d'inventer et d'optimiser une nouvelle génération passionnante de dispositifs de mémoire moléculaire non-volatile («memristive nanodevices»), sont une solution possible à un ensemble d'approches ayant des impacts électroniques, industriels, voire environnementaux. Nous sommes enthousiasmés par l'émergence d'une nouvelle génération d'ordinateurs à mémoire chimique qui peuvent être réalisés non seulement dans les installations de nanofabrication mais aussi dans les espaces de production ou les écoles du monde entier et nous espérons que les leçons et les idées que nous avons obtenues sur ces dispositifs, ainsi que les systèmes peuvent être utiles aux concepteurs et ingénieurs travaillant sur des tâches similaires dans les années à venir.

Bibliography

[1] G. Cook D. Pomeranz and K. Rohrbach. "Click Clean Scorecard: Key Findings and Scores Explained". Technical report, Greenpeace International, 2015. Available: http://www.greenpeace.org/usa/wp-content/uploads/2015/08/2015ClickCleanScorecardKeyFindings.pdf.

[2] K. Traber H. Price A. Horvath W. W. Nazaroff A. Shehabi, S. Ganguly and A. J. Gadgil. "Energy Implications of Economizer Use in California Data Centers". Technical report, Lawrence Berkeley National Laboratory, 2008. Available: https://eta.lbl.gov/publications/energy-implications-economizer-use.

[3] "Gadgets and Gigawatts: Policies for Energy Efficient Electronics". Technical report, International Energy Agency (IEA), 2009. Available: http://www.iea.org/publications/freepublications/publication/gigawatts2009.pdf.

[4] G.-K. Plattner T F. Stocker, Q. Dahe. "Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change (IPCC)". Technical report, Intergovernmental Panel on Climate Change (IPCC), 2013. Available: http://www.climatechange2013.org/report/full-report.

[5] J. H. Faghmous, V. Kumar, and S. Shekhar. Computing and climate. *Computing in Science Engineering*, 17(6):6–8, 2015.

[6] M. Hung. "IoT's Challenges and Opportunities in 2017". Technical report, Gartner Inc, 2017. Available: http://www.gartner.com/technology/research/internet-of-things/report/.

[7] T. Friedman E. Thoo. " IoT Data Proliferation Elevates Data Integration Challenges". Technical report, Gartner Inc, 2016. Available: https://www.gartner.com/doc/3221917.

[8] Blesson Varghese and Rajkumar Buyya. Next generation cloud computing: New trends and research directions. *arXiv preprint arXiv:1707.07452*, 2017.

[9] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets:

A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[10] Robert H Dennard, Fritz H Gaensslen, V Leo Rideout, Ernest Bassous, and Andre R LeBlanc. Design of ion-implanted mosfet's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, 9(5):256–268, 1974.

[11] "2015 International Technology Roadmap for Semiconductors (ITRS)". Technical report, Semiconductor Industry Association, 2015. Available: http://www.itrs2.net/itrs-reports.html.

[12] Laszlo B Kish. End of moore's law: thermal (noise) death of integration in micro and nano electronics. *Physics Letters A*, 305(3):144–149, 2002.

[13] Lei Liao, Yung-Chen Lin, Mingqiang Bao, Rui Cheng, Jingwei Bai, Yuan Liu, Yongquan Qu, Kang L Wang, Yu Huang, and Xiangfeng Duan. High speed graphene transistors with a self-aligned nanowire gate. *Nature*, 467(7313):305, 2010.

[14] Adrian Bachtold, Peter Hadley, Takeshi Nakanishi, and Cees Dekker. Logic circuits with carbon nanotube transistors. *Science*, 294(5545):1317–1320, 2001.

[15] John Von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993.

[16] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, 2014.

[17] John Backus. Can programming be liberated from the von neumann style?: a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641, 1978.

[18] Wei Lu and Charles M Lieber. Nanoelectronics from the bottom up. *Nature materials*, 6(11):841, 2007.

[19] Thomas Rueckes, Kyoungha Kim, Ernesto Joselevich, Greg Y Tseng, Chin-Li Cheung, and Charles M Lieber. Carbon nanotube-based nonvolatile random access memory for molecular computing. *science*, 289(5476):94–97, 2000.

[20] Daniel Gutierrez. "Inside Big Data Guide to In-Memory Computing". Technical report, GridGain, 2014. Available: https://www.gridgain.com/wp-content/uploads/2014/09/insideBIGDATA-Guide-to-In-Memory-Computing.pdf.

[21] H-S Philip Wong and Sayeef Salahuddin. Memory leads the way to better computing. *Nature nanotechnology*, 10(3):191–194, 2015.

[22] Y. V. Pershin and M. Di Ventra. Neuromorphic, digital, and quantum computation with memory circuit elements. *Proceedings of the IEEE*, 100(6):2071–2080, June 2012.

[23] Sasitharan Balasubramaniam and Jussi Kangasharju. Realizing the internet of nano things: challenges, solutions, and applications. *Computer*, 46(2):62–68, 2013.

[24] Crossbar Inc. Crossbar ReRAM Technology White Paper. Technical report, Crossbar Inc., 01 2017.

[25] Introducing intel® optane™ technology – bringing 3d xpoint™ memory to storage and memory products. https://newsroom.intel.com/press-kits/introducing-intel-optane-technology-bringing-3d-xpoint-memory-to-storage-and-memory-products/. Accessed: 2017-09-30.

[26] Leon Chua, Valery Sbitnev, and Hyongsuk Kim. Hodgkin–Huxley Axon Is Made of Memristors. *Int. J. Bifurc. Chaos*, 22(3):1230011, 2012.

[27] Sung Hyun Jo, Ting Chang, Idongesit Ebong, Bhavitavya B Bhadviya, Pinaki Mazumder, and Wei Lu. Nanoscale memristor device as synapse in neuromorphic systems. *Nano letters*, 10(4):1297–1301, 2010.

[28] Geoffrey W. Burr, Robert M. Shelby, Abu Sebastian, Sangbum Kim, Seyoung Kim, Severin Sidler, Kumar Virwani, Masatoshi Ishii, Pritish Narayanan, Alessandro Fumarola, Lucas L. Sanches, Irem Boybat, Manuel Le Gallo, Kibong Moon, Jiyoo Woo, Hyunsang Hwang, and Yusuf Leblebici. Neuromorphic computing using non-volatile memory. *Advances in Physics: X*, 2(1):89–124, 2017.

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

[30] Giacomo Indiveri, Bernabé Linares-Barranco, and Teresa Serrano-Gotarredona. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5(73), 2011.

[31] Shih-Chii Liu. *Analog VLSI: circuits and principles*. 2002.

[32] Carver Mead and Mohammed Ismail. *Analog VLSI implementation of neural systems*, volume 80. Springer Science & Business Media, 2012.

[33] Srinjoy Mitra, Stefano Fusi, and Giacomo Indiveri. Real-time classification of complex patterns using spike-based learning in neuromorphic vlsi. *Biomedical Circuits and Systems, IEEE Transactions on*, 3(1):32–42, 2009.

[34] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V Arthur, Paul A Merolla, and Kwabena Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014.

[35] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in neuroscience*, 9, 2015.

[36] Massimiliano Giulioni, Federico Corradi, Vittorio Dante, and Paolo Del Giudice. Real time unsupervised learning of visual stimuli in neuromorphic vlsi systems. *Scientific reports*, 5:14730, 2015.

[37] Eustace Painkras, Luis A Plana, Jim Garside, Steve Temple, Francesco Galluppi, Cameron Patterson, David R Lester, Andrew D Brown, and Steve B Furber. Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8):1943–1953, 2013.

[38] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557, 2015.

[39] Steve K Esser, Alexander Andreopoulos, Rathinakumar Appuswamy, Pallab Datta, Davis Barch, Arnon Amir, John Arthur, Andrew Cassidy, Myron Flickner, Paul Merolla, et al. Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE, 2013.

[40] Jennifer Hasler and Harry Marr. Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in Neuroscience*, 7:118, 2013.

[41] Giacomo Indiveri, Bernabé Linares-Barranco, Robert Legenstein, George Deligeorgis, and Themistoklis Prodromakis. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, 24(38):384010, 2013.

[42] Ting Chang, Yuchao Yang, and Wei Lu. Building neuromorphic circuits with memristive devices. *IEEE Circuits and Systems Magazine*, 13(2):56–73, 2013.

[43] Konstantin Likharev, Andreas Mayr, Ibrahim Muckra, and Özgür Türel. Crossnets: High-performance neuromorphic architectures for cmol circuits. *Annals of the New York Academy of Sciences*, 1006(1):146–163, 2003.

[44] Yong Chen, Gun-Young Jung, Douglas AA Ohlberg, Xuema Li, Duncan R Stewart, Jan O Jeppesen, Kent A Nielsen, J Fraser Stoddart, and R Stanley Williams. Nanoscale molecular-switch crossbar circuits. *Nanotechnology*, 14(4):462, 2003.

[45] Dmitri B Strukov and Konstantin K Likharev. Cmol fpga: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. *Nanotechnology*, 16(6):888, 2005.

[46] Konstantin K Likharev. Crossnets: Neuromorphic hybrid cmos/nanoelectronic networks. *Science of Advanced Materials*, 3(3):322–331, 2011.

[47] Leon Chua. Memristor-the missing circuit element. *IEEE Transactions on circuit theory*, 18(5):507–519, 1971.

[48] James M Tour and Tao He. The fourth element. *Nature*, 453(7191):42–43, 2008.

[49] Dmitri B Strukov, Gregory S Snider, Duncan R Stewart, and R Stanley Williams. The missing memristor found. *Nature*, 453(7191):80–83, 2008.

[50] Ilia Valov, Eike Linn, Stefan Tappertzhofen, Sebastian Schmelzer, J Van den Hurk, Florian Lentz, and Rainer Waser. Nanobatteries in redox-based resistive switches require extension of memristor theory. *Nature communications*, 4:1771, 2013.

[51] Leon Chua. If it's pinched it'sa memristor. *Semiconductor Science and Technology*, 29(10):104001, 2014.

[52] Massimiliano Di Ventra and Yuriy V Pershin. On the physical properties of memristive, memcapacitive and meminductive systems. *Nanotechnology*, 24(25):255201, 2013.

[53] Massimiliano Di Ventra and Yuriy V Pershin. Memcomputing: a computing paradigm to store and process information on the same physical platform. *arXiv preprint arXiv:1211.4487*, 2012.

[54] J Joshua Yang, Dmitri B Strukov, and Duncan R Stewart. Memristive devices for computing. *Nature Nanotechnology*, 8(1):13–24, 2013.

[55] Rainer Waser and Masakazu Aono. Nanoionics-based resistive switching memories. *Nature materials*, 6(11):833–840, 2007.

[56] Ilia Valov, Rainer Waser, John R Jameson, and Michael N Kozicki. Electrochemical metallization memories—fundamentals, applications, prospects. *Nanotechnology*, 22(25):254003, 2011.

[57] Shanshan Peng, Fei Zhuge, Xinxin Chen, Xiaojian Zhu, Benlin Hu, Liang Pan, Bin Chen, and Run-Wei Li. Mechanism for resistive switching in an oxide-based electrochemical metallization memory. *Applied Physics Letters*, 100(7):072101, 2012.

[58] Rainer Waser, Daniele Ielmini, Hiro Akinaga, Hisashi Shima, H.-S. Philip Wong, Joshua J. Yang, and Simon Yu. *Introduction to Nanoionic Elements for Information Technology*, pages 1–30. Wiley-VCH Verlag GmbH Co. KGaA, 2016.

[59] D. J. Wouters, R. Waser, and M. Wuttig. Phase-change and redox-based resistive switching memories. *Proceedings of the IEEE*, 103(8):1274–1288, Aug 2015.

[60] Yuchao Yang, Peng Gao, Siddharth Gaba, Ting Chang, Xiaoqing Pan, and Wei Lu. Observation of conducting filament growth in nanoscale resistive memories. *Nature communications*, 3:732, 2012.

[61] Umberto Celano, Ludovic Goux, Attilio Belmonte, Karl Opsomer, Alexis Franquet, Andreas Schulze, Christophe Detavernier, Olivier Richard, Hugo Bender, Malgorzata Jurczak, et al. Three-dimensional observation of the conductive filament in nanoscaled resistive memory devices. *Nano letters*, 14(5):2401–2406, 2014.

[62] Mark Buckwell, Luca Montesi, Stephen Hudziak, Adnan Mehonic, and Anthony J Kenyon. Conductance tomography of conductive filaments in intrinsic silicon-rich silica rram. *Nanoscale*, 7(43):18030–18035, 2015.

[63] Sergiu Clima, Yang Yin Chen, Chao Yang Chen, Ludovic Goux, Bogdan Govoreanu, Robin Degraeve, Andrea Fantini, Malgorzata Jurczak, and Geoffrey Pourtois. First-principles thermodynamics and defect kinetics guidelines for engineering a tailored rram device. *Journal of Applied Physics*, 119(22):225107, 2016.

[64] Simone Raoux, Wojciech Wełnic, and Daniele Ielmini. Phase change materials and their application to nonvolatile memories. *Chemical reviews*, 110(1):240–267, 2009.

[65] Weijie Wang, Desmond Loke, Luping Shi, Rong Zhao, Hongxin Yang, Leong-Tat Law, Lung-Tat Ng, Kian-Guan Lim, Yee-Chia Yeo, Tow-Chong Chong, and Andrea L. Lacaita. Enabling universal memory by overcoming the contradictory speed and stability nature of phase-change materials. *Scientific Reports*, 2:360 EP –, 04 2012.

[66] Yi Li, Yingpeng Zhong, Lei Xu, Jinjian Zhang, Xiaohua Xu, Huajun Sun, and Xiangshui Miao. Ultrafast synaptic events in a chalcogenide memristor. *Scientific reports*, 3, 2013.

[67] Tae Hoon Lee, Desmond Loke, Ke-Jie Huang, Wei-Jie Wang, and Stephen R Elliott. Tailoring transient-amorphous states: Towards fast and power-efficient phase-change memory and neuromorphic computing. *Advanced Materials*, 26(44):7493–7498, 2014.

[68] Simone Raoux, Feng Xiong, Matthias Wuttig, and Eric Pop. Phase change materials and phase change memory. *MRS bulletin*, 39(8):703–710, 2014.

[69] Suock Chung, K-M Rho, S-D Kim, H-J Suh, D-J Kim, H-J Kim, S-H Lee, J-H Park, H-M Hwang, S-M Hwang, et al. Fully integrated 54nm stt-ram with the smallest bit cell dimension for high density memory application. In *Electron Devices Meeting (IEDM), 2010 IEEE International*, pages 12–7. IEEE, 2010.

[70] Weisheng Zhao, Sumanta Chaudhuri, Celso Accoto, Jacques-Olivier Klein, Claude Chappert, and Pascale Mazoyer. Cross-point architecture for spin-transfer torque magnetic random access memory. *IEEE Transactions on Nanotechnology*, 11(5):907–917, 2012.

[71] Wang Kang, Weisheng Zhao, Zhaohao Wang, Jacques-Olivier Klein, Yue Zhang, Djaafar Chabi, Youguang Zhang, Dafiné Ravelosona, and Claude Chappert. An overview of spin-based integrated circuits. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pages 676–683. IEEE, 2014.

[72] Yue Zhang, Weisheng Zhao, Guillaume Prenat, Thibaut Devolder, Jacques-Olivier Klein, Claude Chappert, Bernard Dieny, and Dafiné Ravelosona. Electrical modeling of stochastic spin transfer torque writing in magnetic tunnel junctions for memory and logic applications. *IEEE Transactions on Magnetics*, 49(7):4375–4378, 2013.

[73] Adrien F Vincent, Jérôme Larroque, Nicolas Locatelli, Nesrine Ben Romdhane, Olivier Bichler, Christian Gamrat, Wei Sheng Zhao, Jacques-Olivier Klein, Sylvie Galdin-Retailleau, and Damien Querlioz. Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems. *IEEE transactions on biomedical circuits and systems*, 9(2):166–174, 2015.

[74] Mrigank Sharad, Deliang Fan, and Kaushik Roy. Ultra low power associative computing with spin neurons and resistive crossbar memory. In *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, pages 1–6. IEEE, 2013.

[75] Damir Vodenicarevic, Alice Mizrahi, Nicolas Locatelli, Julie Grollier, and Damien Querlioz. Spintronic nanoscillators for unconventional circuits. In *Circuit Theory and Design (ECCTD), 2017 European Conference on*, pages 1–4. IEEE, 2017.

[76] Alice Mizrahi, Nicolas Locatelli, Romain Lebrun, Vincent Cros, Akio Fukushima, Hitoshi Kubota, Shinji Yuasa, Damien Querlioz, and Julie Grollier. Controlling the phase locking of stochastic magnetic bits for ultra-low power computation. *Scientific reports*, 6, 2016.

[77] André Chanthbouala, Arnaud Crassous, Vincent Garcia, Karim Bouzehouane, Stéphane Fusil, Xavier Moya, Julie Allibe, Bruno Dlubak, Julie Grollier, Stéphane Xavier, et al. Solid-state memories based on ferroelectric tunnel junctions. *Nature nanotechnology*, 7(2):101–104, 2012.

[78] Julie Grollier, Damien Querlioz, and Mark D Stiles. Spintronic nanodevices for bioinspired computing. *Proceedings of the IEEE*, 104(10):2024–2039, 2016.

[79] A Radoi, M Dragoman, and D Dragoman. Memristor device based on carbon nanotubes decorated with gold nanoislands. *Applied Physics Letters*, 99(9):093102, 2011.

[80] Si-Yu Liao and et al. Design and modeling of a neuro-inspired learning circuit using nanotube-based memory devices. *IEEE Trans. Circ. Syst.*, 58(9):2172–2181, Sept 2011.

[81] Karim Gacem, Jean-Marie Retrouvey, Djaafar Chabi, Arianna Filoramo, Weisheng Zhao, Jacques-Olivier Klein, and Vincent Derycke. Neuromorphic function learning with carbon nanotube based synapses. *Nanotechnology*, 24:384013, 2013.

[82] Tae Hee Kim, Eun Young Jang, Nyun Jong Lee, Deung Jang Choi, Kyung-Jin Lee, Jungtak Jang, Jin-sil Choi, Seung Ho Moon, and Jinwoo Cheon. Nanoparticle assemblies as memristors. *Nano letters*, 9(6):2229–2233, 2009.

[83] Fabien Alibart, Stéphane Pleutin, David Guérin, Christophe Novembre, Stéphane Lenfant, Kamal Lmimouni, Christian Gamrat, and Dominique Vuillaume. An organic nanoparticle transistor behaving as a biological spiking synapse. *Advanced Functional Materials*, 20(2):330–337, 2010.

[84] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[85] Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1):3–71, 1988.

[86] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 2(1):230–265, 1937.

[87] Christof Teuscher. *Turing's Connectionism: An Investigation of Neural Network Architectures*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[88] Dina Goldin and Peter Wegner. The interactive nature of computing: Refuting the strong church–turing thesis. *Minds and Machines*, 18(1):17–38, 2008.

[89] Hava Siegelmann. Neural networks and analog computation: beyond the turing limit, 2012.

[90] Hajo Broersma, Susan Stepney, and Goran Wendin. Computability and complexity of unconventional computing devices. *arXiv preprint arXiv:1702.02980*, 2017.

[91] Peter S Eriksson, Ekaterina Perfilieva, Thomas Björk-Eriksson, Ann-Marie Alborn, Claes Nordborg, Daniel A Peterson, and Fred H Gage. Neurogenesis in the adult human hippocampus. *Nature medicine*, 4(11):1313–1317, 1998.

[92] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[93] Nikola Kasabov, Kshitij Dhoble, Nuttapod Nuntalid, and Giacomo Indiveri. Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks*, 41:188–201, 2013.

[94] Michael Margulis and Cha-Min Tang. Temporal integration can readily switch between sublinear and supralinear summation. *Journal of neurophysiology*, 79(5):2809–2813, 1998.

[95] Jeffrey C Magee. Dendritic integration of excitatory synaptic input. *Nature Reviews Neuroscience*, 1(3), 2000.

[96] Romain Daniel Cazé, Mark Humphries, and Boris Gutkin. Passive dendrites enable single neurons to compute linearly non-separable functions. *PLoS computational biology*, 9(2):e1002867, 2013.

[97] Rodney J Douglas, Kevan AC Martin, and David Whitteridge. A canonical microcircuit for neocortex. *Neural computation*, 1(4):480–488, 1989.

[98] Rodrigo Quian Quiroga and Stefano Panzeri. Extracting information from neuronal populations: information theory and decoding approaches. *Nature Reviews Neuroscience*, 10(3):173–185, 2009.

[99] Elie L Bienenstock, Leon N Cooper, and Paul W Munro. Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, 2(1):32–48, 1982.

[100] Chris Eliasmith and Charles H Anderson. Neural engineering. *Massachusetts Institue of Technology*, 2003.

[101] Wensheng Sun and Dennis L Barbour. Rate, not selectivity, determines neuronal population coding accuracy in auditory cortex. *PLoS biology*, 15(11):e2002459, 2017.

[102] Romain Brette. Philosophy of the spike: rate-based vs. spike-based theories of the brain. *Frontiers in systems neuroscience*, 9, 2015.

[103] Johanni Brea and Wulfram Gerstner. Does computational neuroscience need new synaptic learning paradigms? *Current Opinion in Behavioral Sciences*, 11:61–66, 2016.

[104] Peter Dominey, Michael Arbib, and Jean-Paul Joseph. A model of corticostriatal plasticity for learning oculomotor associations and sequences. *Journal of cognitive neuroscience*, 7(3):311–336, 1995.

[105] Brendan J Frey and Nebojsa Jojic. A comparison of algorithms for inference and learning in probabilistic graphical models. *IEEE Transactions on pattern analysis and machine intelligence*, 27(9):1392–1416, 2005.

[106] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[107] David Kappel, Stefan Habenschuss, Robert Legenstein, and Wolfgang Maass. Network plasticity as bayesian inference. *PLoS computational biology*, 11(11):e1004485, 2015.

[108] George Clark. The organization of behavior: A neuropsychological theory. *The Journal of Comparative Neurology*, 93(3):459–460, 1950.

[109] David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. MIT press Cambridge, MA, USA:, 1987.

[110] Guo-qiang Bi and Mu-ming Poo. Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual review of neuroscience*, 24(1):139–166, 2001.

[111] Timothée Masquelier and Simon J Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS computational biology*, 3(2):e31, 2007.

[112] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.

[113] Bernhard Nessler, Michael Pfeiffer, Lars Buesing, and Wolfgang Maass. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS computational biology*, 9(4):e1003037, 2013.

[114] David Kappel, Bernhard Nessler, and Wolfgang Maass. Stdp installs in winner-take-all circuits an online approximation to hidden markov model learning. *PLoS computational biology*, 10(3):e1003511, 2014.

[115] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. 1957.

[116] Albert BJ Novikoff. On convergence proofs for perceptrons. Technical report, STANFORD RESEARCH INST MENLO PARK CALIF, 1963.

[117] Marvin L Minsky and Seymour A Papert. *Perceptrons - Expanded Edition*. 1987.

[118] Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. *IRE WESCON Convention Record*, 4:96–104, 1960.

[119] Bernard Widrow and Michael A Lehr. Perceptrons, adalines, and backpropagation. *Arbib*, 4:719–724, 1995.

[120] Richard S Sutton and Andrew G Barto. Toward a modern theory of adaptive networks: expectation and prediction. *Psychological review*, 88(2):135, 1981.

[121] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[122] Pavel Golik, Patrick Doetsch, and Hermann Ney. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, pages 1756–1760, 2013.

[123] Michael A Nielsen. Neural networks and deep learning, 2015.

[124] Merten Joost and Wolfram Schiffmann. Speeding up backpropagation algorithms by using cross-entropy combined with pattern normalization. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):117–126, 1998.

[125] Bernard Widrow, Aaron Greenblatt, Youngsik Kim, and Dookun Park. The no-prop algorithm: A new learning algorithm for multilayer neural networks. *Neural Networks*, 37:182–188, 2013.

[126] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *Proc. IEEE Int. Joint Conference on Neural Networks*, volume 2, pages 985–990. IEEE, 2004.

[127] Mark D McDonnell, Migel D Tissera, Tony Vladusich, André van Schaik, and Jonathan Tapson. Fast, simple and accurate handwritten digit classification by training shallow neural network classifiers with the 'extreme learning machine'algorithm. *PloS one*, 10(8):e0134254, 2015.

[128] Jonathan Tapson, Greg Cohen, Saeed Afshar, Klaus Stiefel, Yossi Buskila, Tara Hamilton, and André van Schaik. Synthesis of neural networks for spatio-temporal spike pattern recognition and processing. *Frontiers in Neuroscience*, 7:153, 2013.

[129] S Ortín, Miguel C Soriano, L Pesquera, Daniel Brunner, D San-Martín, Ingo Fischer, CR Mirasso, and JM Gutiérrez. A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron. *Scientific reports*, 5, 2015.

[130] Hong-Gui Han, Li-Dan Wang, and Jun-Fei Qiao. Hierarchical extreme learning machine for feedforward neural network. *Neurocomputing*, 128:128–135, 2014.

[131] Wenchao Yu, Fuzhen Zhuang, Qing He, and Zhongzhi Shi. Learning deep representations via extreme learning machines. *Neurocomputing*, 149:308–315, 2015.

[132] Migel D Tissera and Mark D McDonnell. Deep extreme learning machines: supervised autoencoding architecture for classification. *Neurocomputing*, 174:42–49, 2016.

[133] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Spin Glass Theory and Beyond: An Introduction to the Replica Method and Its Applications*, pages 411–415. World Scientific, 1987.

[134] Stephen Grossberg. Contour enhancement, short term memory, and constancies in reverberating neural networks. In *Studies of mind and brain*, pages 332–378. Springer, 1982.

[135] Vishwa Goudar and Dean Buonomano. Encoding sensory and motor patterns as time-invariant trajectories in recurrent neural networks. *arXiv preprint arXiv:1701.00838*, 2017.

[136] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

[137] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[138] Yoshua Bengio Ian Goodfellow and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[139] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[140] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pages 2067–2075, 2015.

[141] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.

[142] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.*, 14(11):2531–2560, 2002.

[143] Herbert Jaeger and Harald Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80, 2004.

[144] David Sussillo and Larry F Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

[145] Gandhi Manjunath and Herbert Jaeger. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural computation*, 25(3):671–696, 2013.

[146] Chrisantha Fernando and Sampsa Sojakka. Pattern recognition in a bucket. In *Advances in Artificial Life*, pages 588–597. Springer, 2003.

[147] Tadashi Yamazaki and Shigeru Tanaka. The cerebellum as a liquid state machine. *Neural Networks*, 20(3):290–297, 2007.

[148] Xavier Hinaut and Peter Ford Dominey. Real-time parallel processing of grammatical structure in the fronto-striatal system: A recurrent network simulation study using reservoir computing. *PloS one*, 8(2):e52946, 2013.

[149] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[150] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[151] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[152] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[153] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[154] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[155] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[156] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, volume 1, page 4, 2011.

[157] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

[158] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[159] Yann LeCun, Corinna Cortes, and Christopher JC Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[160] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian

Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[161] David Verstraeten, Benjamin Schrauwen, Sander Dieleman, Philemon Brakel, Pieter Buteneers, and Dejan Pecevski. Oger: modular learning architectures for large-scale sequential processing. *Journal of Machine Learning Research*, 13(Oct):2995–2998, 2012.

[162] Shaji Varghese, Johannes AAW Elemans, Alan E Rowan, and Roeland JM Nolte. Molecular computing: paths to chemical turing machines. *Chemical Science*, 6(11):6050–6058, 2015.

[163] Fabien Alibart, Elham Zamanidoost, and Dmitri B Strukov. Pattern classification by memristive crossbar circuits using ex situ and in situ training. *Nat. Comm.*, 4, 2013.

[164] Sylvain Saighi, Christian G. Mayr, Teresa Serrano-Gotarredona, Heidemarie Schmidt, Gwendal Lecerf, Jean Tomas, Julie Grollier, Sören Boyn, Adrien F. Vincent, Damien Querlioz, Selina La Barbera, Fabien Alibart, Dominique Vuillaume, Olivier Bichler, Christian Gamrat, and Bernabé Linares-Barranco. Plasticity in memristive devices for spiking neural networks. *Frontiers in Neuroscience*, 9, March 2015.

[165] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat. Immunity to Device Variations in a Spiking Neural Network with Memristive Nanodevices. *IEEE Trans. Nanotechnol.*, 12(3):288 – 295, 2013.

[166] Daniele Garbin, Elisa Vianello, Olivier Bichler, Quentin Rafhay, Christian Gamrat, Gérard Ghibaudo, Barbara DeSalvo, and Luca Perniola. Hfo 2-based oxram devices as synapses for convolutional neural networks. *IEEE Transactions on Electron Devices*, 62(8):2494–2501, 2015.

[167] Yuriy V Pershin and Massimiliano Di Ventra. Experimental demonstration of associative memory with memristive neural networks. *Neural Networks*, 23(7):881–886, 2010.

[168] Teresa Serrano-Gotarredona, Timothée Masquelier, Themistoklis Prodromakis, Giacomo Indiveri, and Bernabe Linares-Barranco. Stdp and stdp variations with memristors for spiking neuromorphic learning systems. *Frontiers in neuroscience*, 7:2, 2013.

[169] Sukru B. Eryilmaz, Duygu Kuzum, Rakesh Jeyasingh, SangBum Kim, Matthew BrightSky, Chung Lam, and H.-S. Philip Wong. Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Neuromorphic Engineering*, 8:205, 2014.

[170] Manan Suri, Olivier Bichler, Damien Querlioz, Boubacar Traoré, Olga Cueto, Luca Perniola, Veronique Sousa, Dominique Vuillaume, Christian Gamrat, and Barbara DeSalvo. Physical aspects of low power synapses based on phase change memory devices. *Journal of Applied Physics*, 112(5):054904, 2012.

[171] M. Suri, D. Querlioz, O. Bichler, G. Palma, E. Vianello, D. Vuillaume, C. Gamrat, and B. DeSalvo. Bio-Inspired Stochastic Computing Using Binary CBRAM Synapses. *IEEE Transactions on Electron Devices*, 60(7):2402–2409, 2013.

[172] Damien Querlioz, Olivier Bichler, Adrien Francis Vincent, and Christian Gamrat. Bioinspired programming of memory devices for implementing an inference engine. *Proceedings of the IEEE*, 103(8):1398–1416, 2015.

[173] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

[174] Erika Covi, Stefano Brivio, Alexander Serb, Themis Prodromakis, Marco Fanciulli, and Sabina Spiga. Analog memristive synapse in spiking networks implementing unsupervised learning. *Frontiers in Neuroscience*, 10:482, 2016.

[175] Stefano Ambrogio, Nicola Ciocchini, Mario Laudato, Valerio Milo, Agostino Pirovano, Paolo Fantini, and Daniele Ielmini. Unsupervised learning by spike timing dependent plasticity in phase change memory (pcm) synapses. *Frontiers in neuroscience*, 10, 2016.

[176] Chenchen Liu, Bonan Yan, Chaofei Yang, Linghao Song, Zheng Li, Beiye Liu, Yiran Chen, Hai Li, Qing Wu, and Hao Jiang. A spiking neuromorphic design with resistive crossbar. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pages 1–6. IEEE, 2015.

[177] Fabien Alibart, Stéphane Pleutin, Olivier Bichler, Christian Gamrat, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Dominique Vuillaume. A memristive nanoparticle/organic hybrid synapstor for neuroinspired computing. *Advanced Functional Materials*, 22(3):609–616, 2012.

[178] Selina La Barbera, Adrien F Vincent, Dominique Vuillaume, Damien Querlioz, and Fabien Alibart. Interplay of multiple synaptic plasticity features in filamentary memristive devices for neuromorphic computing. *Scientific reports*, 6:39216, 2016.

[179] Robert Urbanczik and Walter Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528, 2014.

[180] Elham Zamanidoost, Farnood M Bayat, Dmitri Strukov, and Irina Kataeva. Manhattan rule training for memristive crossbar circuit pattern classifiers. In *Intelligent Signal Processing (WISP), 2015 IEEE 9th International Symposium on*, pages 1–6. IEEE, 2015.

[181] Mirko Prezioso, Farnood Merrikh-Bayat, Brian Hoskins, Gina Adam, Konstantin K Likharev, and Dmitri B Strukov. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521:61–64, 2015.

[182] Myonglae Chu, Byoungho Kim, Sangsu Park, Hyunsang Hwang, Moongu Jeon, Byoung Hun Lee, and Byung-Geun Lee. Neuromorphic hardware system for visual pattern recognition with memristor array and CMOS neuron. *IEEE Transactions on Industrial Electronics*, 62(4):2410–2419, 2015.

[183] Peng Yao, Huaqiang Wu, Bin Gao, Sukru Burc Eryilmaz, Xueyao Huang, Wenqiang Zhang, Qingtian Zhang, Ning Deng, Luping Shi, H-S Philip Wong, et al. Face classification using electronic synapses. *Nature Communications*, 8, 2017.

[184] G.W. Burr, R.M. Shelby, C. di Nolfo, J.W. Jang, R.S. Shenoy, P. Narayanan, K. Virwani, E.U. Giacometti, B. Kurdi, and H. Hwang. Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element. In *2014 IEEE International Electron Devices Meeting (IEDM)*, pages 29.5.1–29.5.4, 2014.

[185] Geoffrey W Burr, Robert M Shelby, Severin Sidler, Carmelo Di Nolfo, Junwoo Jang, Irem Boybat, Rohit S Shenoy, Pritish Narayanan, Kumar Virwani, Emanuele U Giacometti, et al. Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices*, 62(11):3498–3507, 2015.

[186] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[187] Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

[188] Shinhyun Choi, Jong Hoon Shin, Jihang Lee, Patrick Sheridan, and Wei D Lu. Experimental demonstration of feature extraction and dimensionality reduction using memristor networks. *Nano Letters*, 17(5):3113–3118, 2017.

[189] Christopher J Rozell, Don H Johnson, Richard G Baraniuk, and Bruno A Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 20(10):2526–2563, 2008.

[190] Patrick M Sheridan, Fuxi Cai, Chao Du, Wen Ma, Zhengya Zhang, and Wei D Lu. Sparse coding with memristor networks. *Nature nanotechnology*, 12(8):784, 2017.

[191] S. G. Hu, Y. Liu, Z. Liu, T. P. Chen, J. J. Wang, Q. Yu, L. J. Deng, Y. Yin, and Sumio Hosaka. Associative memory realized by a reconfigurable memristive hopfield neural network. *Nature Communications*, 6:7522, 2015.

[192] Yun Long, Eui Min Jung, Jaeha Kung, and Saibal Mukhopadhyay. Reram crossbar based recurrent neural network for human activity detection. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 939–946. IEEE, 2016.

[193] Manjari S Kulkarni and Christof Teuscher. Memristor-based reservoir computing. In *Nanoscale Architectures (NANOARCH), 2012 IEEE/ACM International Symposium on*, pages 226–232. IEEE, 2012.

[194] Dhireesha Kudithipudi, Qutaiba Saleh, Cory Merkel, James Thesing, and Bryant Wysocki. Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing. *Frontiers in neuroscience*, 9, 2015.

[195] Jens Bürger, Alireza Goudarzi, Darko Stefanovic, and Christof Teuscher. Hierarchical composition of memristive networks for real-time computing. In *Nanoscale Architectures (NANOARCH), 2015 IEEE/ACM International Symposium on*, pages 33–38. IEEE, 2015.

[196] Zoran Konkoli. On reservoir computing: from mathematical foundations to unconventional applications. In *Advances in Unconventional Computing*, pages 573–607. Springer, 2017.

[197] Juan Pablo Carbajal, Joni Dambre, Michiel Hermans, and Benjamin Schrauwen. Memristor models for machine learning. *Neural computation*, 2015.

[198] Christopher Bennett, Aldo Jesorka, Göran Wendin, and Zoran Konkoli. On the inverse pattern recognition problem in the context of the time-series data processing with memristor networks. In *Advances in Unconventional Computing*, pages 735–757. Springer, 2017.

[199] Kuk-Hwan Kim, Siddharth Gaba, Dana Wheeler, Jose M Cruz-Albrecht, Tahir Hussain, Narayan Srinivasa, and Wei Lu. A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications. *Nano letters*, 12(1):389–395, 2011.

[200] B Govoreanu, GS Kar, YY Chen, V Paraschiv, S Kubicek, A Fantini, IP Radu, L Goux, S Clima, R Degraeve, et al. 10× 10nm 2 hf/hfo x crossbar resistive ram with excellent performance, reliability and low-energy operation. In *Electron Devices Meeting (IEDM), 2011 IEEE International*, pages 31–6. IEEE, 2011.

[201] Stefano Ambrogio, Simone Balatti, Valerio Milo, Roberto Carboni, Zhong-Qiang Wang, Alessandro Calderoni, Nirmal Ramaswamy, and Daniele Ielmini. Neuromorphic learning and recognition with one-transistor-one-resistor synapses and bistable metal oxide rram. *IEEE Transactions on Electron Devices*, 63(4):1508–1515, 2016.

[202] Myoung-Jae Lee, Dongsoo Lee, Seong-Ho Cho, Ji-Hyun Hur, Sang-Moon Lee, David H Seo, Dong-Sik Kim, Moon-Seung Yang, Sunghun Lee, Euichul Hwang, et al. A plasma-

treated chalcogenide switch device for stackable scalable 3d nanoscale memory. *Nature communications*, 4:2629, 2013.

[203] Jiale Liang and HS Philip Wong. Cross-point memory array without cell selectors—device characteristics and data storage pattern dependencies. *IEEE Transactions on Electron Devices*, 57(10):2531–2538, 2010.

[204] Djaafar Chabi, Damien Querlioz, Weisheng Zhao, and Jacques-Olivier Klein. Robust learning approach for neuro-inspired nanoscale crossbar architecture. *J. Emerg. Technol. Comput. Syst.*, 10(1):5:1–5:20, January 2014.

[205] J Joshua Yang, M-X Zhang, Matthew D Pickett, Feng Miao, John Paul Strachan, Wen-Di Li, Wei Yi, Douglas AA Ohlberg, Byung Joon Choi, Wei Wu, et al. Engineering nonlinearity into memristors for passive crossbar applications. *Applied Physics Letters*, 100(11):113501, 2012.

[206] Eike Linn, Roland Rosezin, Carsten Kügeler, and Rainer Waser. Complementary resistive switches for passive nanocrossbar memories. *Nature materials*, 9(5):403–406, 2010.

[207] E Linn, S Menzel, S Ferch, and R Waser. Compact modeling of crs devices based on ecm cells for memory, logic and neuromorphic applications. *Nanotechnology*, 24(38):384008, 2013.

[208] Sung Hyun Jo, Tanmay Kumar, Sundar Narayanan, Wei D Lu, and Hagop Nazarian. 3d-stackable crossbar resistive memory based on field assisted superlinear threshold (fast) selector. In *IEDM Tech. Dig.*, pages 6–7. IEEE, 2014.

[209] Adam Z Stieg, Audrius V Avizienis, Henry O Sillin, Cristina Martin-Olmos, Masakazu Aono, and James K Gimzewski. Emergent criticality in complex turing b-type atomic switch networks. *Advanced Materials*, 24(2):286–293, 2012.

[210] Henry O Sillin, Renato Aguilera, Hsien-Hang Shieh, Audrius V Avizienis, Masakazu Aono, Adam Z Stieg, and James K Gimzewski. A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing. *Nanotechnology*, 24(38):384004, 2013.

[211] Jens Burger, Alireza Goudarzi, Darko Stefanovic, and Christof Teuscher. Computational capacity and energy consumption of complex resistive switch networks. *arXiv preprint arXiv:1507.03716*, 2015.

[212] Kelsey Scharnhorst, Walt Woods, Christof Teuscher, Adam Stieg, and James Gimzewski. Non-temporal logic performance of an atomic switch network. In *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 133–138. IEEE, 2017.

[213] Jens Bürger and Christof Teuscher. Variation-tolerant computing with memristive reservoirs. In *Proceedings of the 2013 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 1–6. IEEE Press, 2013.

[214] Alireza Goudarzi, Matthew R Lakin, Darko Stefanovic, and Christof Teuscher. A model for variation-and fault-tolerant digital logic using self-assembled nanowire architectures. In *Proceedings of the 2014 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 116–121. ACM, 2014.

[215] Djaafar Chabi, Weisheng Zhao, Damien Querlioz, and Jacques Olivier Klein. Robust Neural Logic Block (NLB) based on memristor crossbar array. *Proceedings of the 2011 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 137–143, 2011.

[216] D. Querlioz, O. Bichler, A.F. Vincent, and C. Gamrat. Bioinspired programming of memory devices for implementing an inference engine. *Proceedings of the IEEE*, 103(8):1398–1416, Aug 2015.

[217] Djaafar Chabi. *Architectures de circuits nanoélectroniques neuro-inspirée*. PhD thesis, Paris 11, 2012.

[218] I. Kataeva, F. Merrikh-Bayat, E. Zamanidoost, and D. Strukov. Efficient training algorithms for neural networks based on memristive crossbar circuits. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2015.

[219] Ligang Gao, Fabien Alibart, and Dmitri B Strukov. Analog-input analog-weight dot-product operation with ag/a-si/pt memristive devices. In *VLSI and System-on-Chip, 2012 (VLSI-SoC), IEEE/IFIP 20th International Conference on*, pages 88–93. IEEE, 2012.

[220] Fabien Alibart, Ligang Gao, Brian D Hoskins, and Dmitri B Strukov. High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. *Nanotechnology*, 23(7):075201, 2012.

[221] D. Chabi, Zhaohao Wang, Weisheng Zhao, and J.-O. Klein. On-chip supervised learning rule for ultra high density neural crossbar using memristor for synapse and neuron. In *IEEE/ACM Int. Symp. Nanoscale Architectures (NANOARCH)*, pages 7–12, July 2014.

[222] Df Tank and John J Hopfield. Simple'neural'optimization networks: An a/d converter, signal decision circuit, and a linear programming circuit. *IEEE Trans. Circ. Syst.*, 33(5):533–541, 1986.

[223] Anirban Bandyopadhyay and Amlan J Pal. Large conductance switching and memory effects in organic molecules for data-storage applications. *Applied physics letters*, 82(8):1215–1217, 2003.

[224] Youngjun Park and Jang-Sik Lee. Artificial synapses with short-and long-term memory for spiking neural networks based on renewable materials. *ACS nano*, 2017.

[225] Nazrin Kooy, Khairudin Mohamed, Lee Tze Pin, and Ooi Su Guan. A review of roll-to-roll nanoimprint lithography. *Nanoscale Research Letters*, 9(1):1–13, 2014.

[226] Valerio Zardetto, Thomas M Brown, Andrea Reale, and Aldo Di Carlo. Substrates for flexible electronics: A practical investigation on the electrical, film flexibility, optical, temperature, and solvent resistance properties. *Journal of Polymer Science Part B: Polymer Physics*, 49(9):638–648, 2011.

[227] Sunghoon Song, Byungjin Cho, Tae-Wook Kim, Yongsung Ji, Minseok Jo, Gunuk Wang, Minhyeok Choe, Yung Ho Kahng, Hyunsang Hwang, and Takhee Lee. Three-dimensional integration of organic resistive memory devices. *Advanced Materials*, 22(44):5048–5052, 2010.

[228] Robert M Metzger. Unimolecular electronics. *Journal of Materials Chemistry*, 18(37):4364–4396, 2008.

[229] Luisa Torsi, Maria Magliulo, Kyriaki Manoli, and Gerardo Palazzo. Organic field-effect transistor sensors: a tutorial review". *Chem. Soc. Rev.*, 42:8612–8628, 2013.

[230] Lei Wang, CiHui Yang, Jing Wen, Shan Gai, and YuanXiu Peng. Overview of emerging memristor families from resistive memristor to spintronic memristor. *Journal of Materials Science: Materials in Electronics*, 26(7):4618–4628, 2015.

[231] J Campbell Scott and Luisa D Bozano. Nonvolatile memory elements based on organic materials. *Advanced Materials*, 19(11):1452–1463, 2007.

[232] Yu Chen, Gang Liu, Cheng Wang, Wenbin Zhang, Run-Wei Li, and Luxing Wang. Polymer memristor for information storage and neuromorphic applications. *Materials Horizons*, 1(5):489–506, 2014.

[233] V. A. Demin, V. V. Erokhin, A. V. Emelyanov, S. Battistoni, G. Baldi, S. Iannotta, P. K. Kashkarov, and M. V. Kovalchuk. Hardware elementary perceptron based on polyaniline memristive devices. *Organic Electronics*, 25:16–20, 2015. not bad?

[234] Victor Erokhin, Tatiana Berzina, Anteo Smerieri, Paolo Camorani, Svetlana Erokhina, and MP Fontana. Bio-inspired adaptive networks based on organic memristors. *Nano Communication Networks*, 1(2), 2010.

[235] Paul Heremans, Gerwin H Gelinck, Robert Muller, Kang-Jun Baeg, Dong-Yu Kim, and Yong-Young Noh. Polymer and organic nonvolatile memory devices. *Chemistry of Materials*, 23(3):341–358, 2010.

[236] Takhee Lee and Yong Chen. Organic resistive nonvolatile memory materials. *MRS Bulletin*, 37:144–149, 2 2012.

[237] Anr-moorea: Memristors organiques et apprentisage. http://www.anr-moorea.u-psud.fr/. Accessed: 2017-10-15.

[238] Théo Cabaret. *Etude, réalisation et caractérisation de memristors organiques électro-greffés en tant que nanosynapses de circuits neuro-inspirés*. PhD thesis, Paris 11, 2014.

[239] T Cabaret, L Fillaud, B Jousselme, JO Klein, and V Derycke. Electro-grafted organic memristors: Properties and prospects for artificial neural networks based on stdp. In *Int. Conf. Nanotechnology (IEEE-NANO)*, pages 499–504, 2014.

[240] Bruno Jousselme, Gérard Bidan, Martial Billon, Cédric Goyer, Yann Kervella, Stéphane Guillerez, Edy Abou Hamad, Christophe Goze-Bac, Jean-Yves Mevellec, and Serge Lefrant. One-step electrochemical modification of carbon nanotubes by ruthenium complexes via new diazonium salts. *Journal of Electroanalytical Chemistry*, 621(2):277 – 285, 2008. Special Issue in Honor of Professor Israel Rubinstein.

[241] Kannan Balasubramanian, Roman Sordan, Marko Burghard, and Klaus Kern. A selective electrochemical approach to carbon nanotube field-effect transistors. *Nano Letters*, 4(5):827–830, 2004.

[242] Julienne Charlier, Laurent Baraton, Christophe Bureau, and Serge Palacin. Directed organic grafting on locally doped silicon substrates. *ChemPhysChem*, 6(1):70–74, 2005.

[243] Kyooho Jung, Yongmin Kim, Young S Park, Woong Jung, Jungae Choi, Baeho Park, Hyungsang Kim, Wondong Kim, Jinpyo Hong, and Hyunsik Im. Unipolar resistive switching in insulating niobium oxide film and probing electroforming induced metallic components. *Journal of Applied Physics*, 109(5):054511, 2011.

[244] Un-Bin Han and Jang-Sik Lee. Integration scheme of nanoscale resistive switching memory using bottom-up processes at room temperature for high-density memory applications. *Scientific reports*, 6:28966, 2016.

[245] Djaafar Chabi, Weisheng Zhao, Damien Querlioz, and Jacques-Olivier Klein. On-chip universal supervised learning methods for neuro-inspired block of memristive nanodevices. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 11(4):34, 2015.

[246] Philip J Kuekes, Duncan R Stewart, and R Stanley Williams. The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits. *Journal of Applied Physics*, 97(3):034301, 2005.

[247] Ahmed G Radwan and Mohammed E Fouda. *On the mathematical modeling of memristor, memcapacitor, and meminductor*, volume 26. Springer, 2015.

[248] Alon Ascoli, Fernando Corinto, Vanessa Senger, and Ronald Tetzlaff. Memristor model comparison. *IEEE Circuits and Systems Magazine*, 13(2):89–105, 2013.

[249] John Paul Strachan, Antonio C Torrezan, Feng Miao, Matthew D Pickett, J Joshua Yang, Wei Yi, Gilberto Medeiros-Ribeiro, and R Stanley Williams. State dynamics and modeling of tantalum oxide memristors. *IEEE Transactions on Electron Devices*, 60(7):2194–2202, 2013.

[250] Lu Zhang, Zhijie Chen, J Joshua Yang, Bryant Wysocki, Nathan McDonald, and Yiran Chen. A compact modeling of tio2-tio2–x memristor. *Applied Physics Letters*, 102(15):153503, 2013.

[251] Michihito Ueda, Yu Nishitani, Yukihiro Kaneko, and Atsushi Omote. Back-propagation operation for analog neural network hardware with synapse components having hysteresis characteristics. *PloS one*, 9(11):e112659, 2014.

[252] G Foffani, ML Morales-Botello, and J Aguilar. Spike timing, spike count, and temporal information for the discrimination of tactile stimuli in the rat ventrobasal complex. *Journal of Neuroscience*, 29(18):5964–5973, 2009.

[253] Alexander Serb, Johannes Bill, Ali Khiat, Radu Berdan, Robert Legenstein, and Themis Prodromakis. Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses. *Nature Communications*, 7:12611, 2016.

[254] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[255] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[256] Robert M Shelby, Geoffrey W Burr, Irem Boybat, and Carmelo di Nolfo. Non-volatile memory as hardware synapse in neuromorphic computing: A first look at reliability issues. In *Reliability Physics Symposium (IRPS), 2015 IEEE International*, pages 6A–1. IEEE, 2015.

[257] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[258] Djaafar Chabi, Zhaohao Wang, Christopher Bennett, Jacques-Olivier Klein, and Weisheng Zhao. Ultrahigh density memristor neural crossbar for on-chip supervised learning. *Nanotechnology, IEEE Transactions on*, 14(6):954–962, 2015.

[259] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[260] Pai-Yu Chen, Binbin Lin, I-Ting Wang, Tuo-Hung Hou, Jieping Ye, Sarma Vrudhula, Jae-sun Seo, Yu Cao, and Shimeng Yu. Mitigating effects of non-ideal synaptic device characteristics for on-chip learning. In *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*, pages 194–199. IEEE, 2015.

[261] Wouter F Schmidt, Martin A Kraaijveld, and Robert PW Duin. Feedforward neural networks with random weights. In *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, pages 1–4. IEEE, 1992.

[262] Panayiota Poirazi, Terrence Brannon, and Bartlett W Mel. Pyramidal neuron as two-layer neural network. *Neuron*, 37(6):989–999, 2003.

[263] M. Suri and V. Parmar. Exploiting intrinsic variability of filamentary resistive memory for extreme learning machine architectures. *IEEE Trans. Nanotechnol.*, 14(6):963–968, 2015.

[264] Manan Suri, Vivek Parmar, Gilbert Sassine, and Fabien Alibart. Oxram based elm architecture for multi-class classification applications. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[265] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[266] Damien Querlioz, Philippe Dollfus, Olivier Bichler, and Christian Gamrat. Learning with memristive devices: How should we model their behavior? In *Proceedings of the 2011 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 150–156. IEEE Computer Society, 2011.

[267] Wolfgang Maass. Neural computation with winner-take-all as the only nonlinear operation. In *Advances in neural information processing systems*, pages 293–299, 2000.

[268] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[269] D. Soudry, D. Di Castro, A. Gal, A. Kolodny, and S. Kvatinsky. Memristor-based multilayer neural networks with online gradient descent training. *IEEE Trans. Neural Netw., in press*, 2015.

[270] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pages 1–5. IEEE, 2015.

[271] Olivier Bichler, Manan Suri, Damien Querlioz, Dominique Vuillaume, Barbara DeSalvo, and Christian Gamrat. Visual pattern extraction using energy-efficient "2-pcm synapse" neuromorphic architecture. *IEEE Transactions on Electron Devices*, 59(8):2206–2214, 2012.

[272] Geoffrey W Burr, Rohit S Shenoy, Kumar Virwani, Pritish Narayanan, Alvaro Padilla, Bülent Kurdi, and Hyunsang Hwang. Access devices for 3d crosspoint memory a. *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, 32(4):040802, 2014.

[273] Yoeri van de Burgt, Ewout Lubberman, Elliot J Fuller, Scott T Keene, Grégorio C Faria, Sapan Agarwal, Matthew J Marinella, A Alec Talin, and Alberto Salleo. A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing. *Nature Materials*, 16(4):414–418, 2017.

[274] André Chanthbouala, Vincent Garcia, Ryan O Cherifi, Karim Bouzehouane, Stéphane Fusil, Xavier Moya, Stéphane Xavier, Hiroyuki Yamada, Cyrile Deranlot, Neil D Mathur, et al. A ferroelectric memristor. *Nature materials*, 11(10):860–864, 2012.

[275] Thomas Pfeil, Tobias Potjans, Sven Schrader, Wiebke Potjans, Johannes Schemmel, Markus Diesmann, and Karlheinz Meier. Is a 4-bit synaptic weight resolution enough? – constraints on enabling spike-timing dependent plasticity in neuromorphic hardware. *Frontiers in Neuroscience*, 6:90, 2012.

[276] Walt Woods, Jens Bürger, and Christof Teuscher. Synaptic weight states in a locally competitive algorithm for neuromorphic memristive hardware. *IEEE Transactions on Nanotechnology*, 14(6):945–953, 2015.

[277] Elham Zamanidoost, Michael Klachko, Dmitri Strukov, and Irina Kataeva. Low area overhead in-situ training approach for memristor-based classifier. In *IEEE/ACM Int. Symp. Nanoscale Architectures (NANOARCH), 2015*, pages 139–142. IEEE, 2015.

[278] Tayfun Gokmen and Yurii Vlasov. Acceleration of deep neural network training with resistive cross-point devices: Design considerations. *Frontiers in Neuroscience*, 10:333, 2016.

[279] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[280] Shinhyun Choi, Patrick Sheridan, and Wei D Lu. Data clustering using memristor networks. *Scientific reports*, 5, 2015.

[281] Kyriaki Sidiropoulou and Panayiota Poirazi. Predictive features of persistent activity emergence in regular spiking and intrinsic bursting model neurons. *PLoS Comput Biol*, 8(4):e1002489, 2012.

[282] Stephen A Baccus and Markus Meister. Fast and slow contrast adaptation in retinal circuitry. *Neuron*, 36(5):909–919, 2002.

[283] Nicole C Rust, Odelia Schwartz, J Anthony Movshon, and Eero P Simoncelli. Spatiotemporal elements of macaque v1 receptive fields. *Neuron*, 46(6):945–956, 2005.

[284] Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

[285] Niru Maheswaranathan, Stephen A Baccus, and Surya Ganguli. Inferring hidden structure in multilayered neural circuits. *bioRxiv*, page 120956, 2017.

[286] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[287] Henry Markram, Dimitri Pikus, Anirudh Gupta, and Misha Tsodyks. Potential for multiple mechanisms, phenomena and algorithms for synaptic plasticity at single synapses. *Neuropharmacology*, 37(4):489–500, 1998.

[288] Wickliffe C Abraham. Metaplasticity: tuning synapses and networks for plasticity. *Nature Reviews Neuroscience*, 9(5), 2008.

[289] Endel Tulving. How many memory systems are there? *American psychologist*, 40(4):385, 1985.

[290] Endel Tulving, Daniel L Schacter, et al. Priming and human memory systems. *Science*, 247(4940):301–306, 1990.

[291] Ting Chang, Sung-Hyun Jo, and Wei Lu. Short-term memory to long-term memory transition in a nanoscale memristor. *ACS nano*, 5(9):7669–7676, 2011.

[292] Zhong Qiang Wang, Hai Yang Xu, Xing Hua Li, Hao Yu, Yi Chun Liu, and Xiao Juan Zhu. Synaptic learning and memory functions achieved using oxygen ion migration/diffusion in an amorphous ingazno memristor. *Adv. Func. Mater.*, 22(13):2759–2765, 2012.

[293] Selina La Barbera, Dominique Vuillaume, and Fabien Alibart. Filamentary switching: Synaptic plasticity through device volatility. *ACS nano*, 9(1):941–949, 2015.

[294] Zheng-Hua Tan, Rui Yang, Kazuya Terabe, Xue-Bing Yin, Xiao-Dong Zhang, and Xin Guo. Synaptic metaplasticity realized in oxide memristive devices. *Advanced Materials*, 2015.

[295] Xiaojian Zhu, Chao Du, YeonJoo Jeong, and Wei D Lu. Emulation of synaptic metaplasticity in memristors. *Nanoscale*, 9(1):45–51, 2017.

[296] Jens Bürger and Christof Teuscher. Volatile memristive devices as short-term memory in a neuromorphic learning architecture. In *Proceedings of the 2014 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 104–109. ACM, 2014.

[297] Jonathan Tapson, Philip de Chazal, and André van Schaik. Explicit computation of input weights in extreme learning machines. In *Proceedings of ELM-2014 Volume 1*, pages 41–49. Springer, 2015.

[298] Runchun Wang, Chetan Singh Thakur, Gregory Cohen, Tara Julia Hamilton, Jonathan Tapson, and André van Schaik. Neuromorphic hardware architecture using the neural engineering framework for pattern recognition. *IEEE Transactions on Biomedical Circuits and Systems*, 11(3):574–584, 2017.

[299] John A Lansner and Torsten Lehmann. An analog cmos chip set for neural networks with arbitrary topologies. *IEEE Transactions on Neural Networks*, 4(3):441–444, 1993.

[300] Thomas M Bartol, Cailey Bromer, Justin P Kinney, Micheal A Chirillo, Jennifer N Bourne, Kristen M Harris, and Terrence J Sejnowski. Nanoconnectomic upper bound on the variability of synaptic plasticity. *eLife*, page e10778, 2015.

[301] Richard Lyon. A computational model of filtering, detection, and compression in the cochlea. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, volume 7, pages 1282–1285. IEEE, 1982.

[302] David Verstraeten, Benjamin Schrauwen, Dirk Stroobandt, and Jan Van Campenhout. Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, 95(6):521–528, 2005.

[303] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[304] Alex Graves, Douglas Eck, Nicole Beringer, and Juergen Schmidhuber. Biologically plausible speech recognition with lstm neural nets. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pages 127–136. Springer, 2004.

[305] Yvan Paquot, Francois Duport, Antoneo Smerieri, Joni Dambre, Benjamin Schrauwen, Marc Haelterman, and Serge Massar. Optoelectronic reservoir computing. *Scientific reports*, 2:287, 2012.

[306] JB Butcher, David Verstraeten, Benjamin Schrauwen, CR Day, and PW Haycock. Reservoir computing and extreme learning machines for non-linear time-series data analysis. *Neural networks*, 38:76–89, 2013.

[307] Guang-Bin Huang, Zuo Bai, Liyanaarachchi Lekamalage Chamara Kasun, and Chi Man Vong. Local receptive fields based extreme learning machine. *IEEE Computational Intelligence Magazine*, 10(2):18–29, 2015.

[308] Miquel L Alomar, Vincent Canals, Nicolas Perez-Mora, Víctor Martínez-Moll, and Josep L Rosselló. Fpga-based stochastic echo state networks for time-series forecasting. *Computational intelligence and neuroscience*, 2016:15, 2016.

[309] Damien Querlioz, WS Zhao, Philippe Dollfus, J-O Klein, Olivier Bichler, and Christian Gamrat. Bioinspired networks with nanoscale memristive devices that combine the unsupervised and supervised learning approaches. In *Proceedings of the 2012 IEEE/ACM International Symposium on Nanoscale Architectures*, pages 203–210. ACM, 2012.

[310] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *Proceedings of the 43rd International Symposium on Computer Architecture*, pages 14–26. IEEE Press, 2016.

[311] L. Ni, Z. Liu, W. Song, J. J. Yang, H. Yu, K. Wang, and Y. Wang. An energy-efficient and high-throughput bitwise cnn on sneak-path-free digital reram crossbar. In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 1–6, July 2017.

[312] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang. Binary convolutional neural network on rram. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 782–787, Jan 2017.

[313] Stephen Marsland. Novelty detection in learning systems. *Neural computing surveys*, 3(2):157–195, 2003.

[314] Tarem Ahmed, Boris Oreshkin, and Mark Coates. Machine learning approaches to network anomaly detection. In *Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, pages 1–6. USENIX Association, 2007.

[315] J-P Halain, Arnaud Debaize, J-M Gillis, Lionel Jacques, Tine De Ridder, Lou Hermans, Manuel Koch, Guy Meynants, and Gert Schippers. The dual-gain 10 μm back-thinned 3k× 3k cmos-aps detector of the solar orbiter extreme uv imager. In *SPIE Astronomical Telescopes+ Instrumentation*, pages 91443I–91443I. International Society for Optics and Photonics, 2014.

[316] T Horiuchi and Ralph Etienne-Cummings. A time-series processor for sonar mapping and novelty detection. In *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*, volume 4, pages IV–868. IEEE, 2003.

[317] Stephen D Smith, Michael J Dixon, William J Tays, and M Barbara Bulman-Fleming. Anomaly detection in the right hemisphere: The influence of visuospatial factors. *Brain and cognition*, 55(3):458–462, 2004.

[318] Vilayanur S Ramachandran. Anosognosia in parietal lobe syndrome. *Consciousness and cognition*, 4(1):22–51, 1995.

[319] Duygu Kuzum, Rakesh GD Jeyasingh, and H-S Philip Wong. Energy efficient programming of nanoelectronic synaptic devices for large-scale implementation of associative and temporal sequence learning. In *Electron Devices Meeting (IEDM), 2011 IEEE International*, pages 30–3. IEEE, 2011.

[320] Jie Han and Michael Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *Test Symposium (ETS), 2013 18th IEEE European*, pages 1–6. IEEE, 2013.

[321] Teuvo Kohonen and Erkki Oja. Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics*, 21(2):85–95, 1976.

[322] T. Kohonen. *Self-organization and Associative Memory: 3rd Edition*. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[323] Ailong Wu, Shiping Wen, and Zhigang Zeng. Synchronization control of a class of memristor-based recurrent neural networks. *Information Sciences*, 183(1):106–116, 2012.

[324] Idongesit E Ebong and Pinaki Mazumder. Cmos and memristor-based neural network design for position detection. *Proceedings of the IEEE*, 100(6):2050–2060, 2012.

[325] Ning Ge, Jung Ho Yoon, Miao Hu, EJ Merced-Grafals, Noraica Davila, John Paul Strachan, Zhiyong Li, Helen Holder, Qiangfei Xia, R Stanley Williams, et al. An efficient analog hamming distance comparator realized with a unipolar memristor array: a showcase of physical computing. *Scientific reports*, 7, 2017.

université PARIS-SACLAY | ÉCOLE DOCTORALE Physique et ingénierie : électrons, photons, sciences du vivant (EOBE)

Titre : Apprentisage local avec des dispositifs de mémoire hautement analogiques

Mots clés : neuromorphique, memristor, nanodispositif, modélisation, nanoelectronique, réseaux de neurones

Résumé : Dans la prochaine ère de l'informatique distribuée, les ordinateurs inspirés par le cerveau qui effectuent des opérations localement plutôt que dans des serveurs distants seraient un avantage majeur en réduisant les coûts énergétiques et réduisant l'impact environnemental. Une nouvelle génération de nanodispositifs de mémoire non-volatile est un candidat de premier plan pour réaliser cette vision neuromorphique. À l'aide de travaux théoriques et expérimentaux, nous avons exploré les problèmes critiques qui se posent lors de la réalisation physique des architectures de réseaux de neurones artificiels modernes (ANN) en utilisant des dispositifs de mémoire émergents (nanodispositifs « memristifs »).
Dans notre travail expérimental, nos dispositifs organiques (polymeriques) se sont adaptés avec succès et automatiquement en tant que portes logiques reconfigurables en coopérant avec un neurone digital et programmable (FGPA).
Dans nos travaux théoriques, nous aussi avons considéré les multicouche memristive ANNs. Nous avons développé et simulé des variantes de projection aléatoire (un système NoProp) et de rétropropagation (un système perceptron multicouche) qui utilisent deux crossbars. Ces systèmes d'apprentissage locaux ont montré des dépendances critiques sur les contraintes physiques des nanodispositifs. Enfin, nous avons examiné comment les conceptions ANNs “feed-forward” peuvent être modi-fiées pour exploiter les effets temporels. Nous avons amélioré la bio-inspiration et la performance du système NoProp, par exemple, avec des effets de plasticité dans la première couche. Ces effets ont été obtenus en utilisant un nanodispositif à ionisation d'argent avec un comportement de transition de plasticité intrinsèque.

Title : Local learning with highly analog memory nanodevices

Keywords : neuromorphic, memristor, nanodevice, compact modeling, nanoelectronics, neural networks

Abstract : In the next era of distributed computing, brain-based computers that perform operations locally rather than in remote servers would be a major benefit in reducing global energy costs. A new generation of emerging nonvolatile memory devices is a leading candidate for achieving this neuromorphic vision. Using theoretical and experimental work, we have explored critical issues that arise when physically realizing modern artificial neural network (ANN) architectures using emerging memory devices (“memristors”). In our experimental work, we showed organic nanosynapses adapting automatically as logic gates via a companion digital neuron and programmable logic cell (FGPA).
In our theoretical work, we also considered multilayer memristive ANNs. We have developed and simulated random projection (NoProp) and backpropagation (Multilayer Perceptron) variants that use two crossbars. These local learning systems showed critical dependencies on the physical constraints of nanodevices. Finally, we examined how feed-forward ANN designs can be modified to exploit temporal effects. We focused in particular on improving bio-inspiration and performance of the NoProp system, for example, we improved the performance with plasticity effects in the first layer. These effects were obtained using a silver ionic nanodevice with intrinsic plasticity transition behavior.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l’Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France