Memristive analog computing and innovative sensors for neuromorphic systems

Filippo Moro

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : EEATS - Electronique, Electrotechnique, Automatique, Traitement du Signal (EEATS)
Spécialité : Nano électronique et Nano technologies
Unité de recherche : CEA/LETI

Calcul analogique memristif et capteurs innovants pour systèmes neuromorphiques

Memristive analog computing and innovative sensors for neuromorphic systems

Présentée par :

Filippo MORO

Direction de thèse :

Julien ARCAMONE Chef du Service CEA/DRT/LETI/DCOS/S3C Université Grenoble Alpes	Directeur de thèse
Elisa VIANELLO CEA/DRT/LETI/DCOS	Co-encadrante de thèse

Rapporteurs :

Damien QUERLIOZ
CHARGE DE RECHERCHE, CNRS
Fabien ALIBART
CHARGE DE RECHERCHE, CNRS

Thèse soutenue à huis clos le **28 mars 2023**, devant le jury composé de :

Julien ARCAMONE DIRECTEUR DE RECHERCHE, CEA	Directeur de thèse
Damien QUERLIOZ CHARGE DE RECHERCHE, CNRS	Rapporteur
Fabien ALIBART CHARGE DE RECHERCHE, CNRS	Rapporteur
Julie GROLLIER DIRECTEUR DE RECHERCHE, CNRS	Examinatrice
Quentin RAFHAY MAITRE DE CONFERENCES, Grenoble-INP	Examinatrice
Marc BOCQUET PROFESSEUR DES UNIVERSITES, Aix Marseille Université	Examinateur
Giorgio DI NATALE DIRECTEUR DE RECHERCHE, CNRS	Président de jury

Invités :

Elisa Vianello
ingénieure docteur, CEA-Leti

Acknowledgments

A thesis has one author only but is teamwork. There are many people to whom I am thankful for collaborating on my projects and many more who have supported me throughout these 3 years.
I want to begin with Elisa, who welcomed me into her lab and has always been there technically and emotionally. I really admire the way you lead the lab, projects, and your strength in supporting your ideas. Thank you so much for being the best possible supervisor for my thesis!
Melika and Thomas, you are my mentors. Most of what I know, its either from or because of you. You trusted me in the very beginning of my journey to take on some of your projects and had the patience to teach me not just technical aspects, but also shared with me your vision. I am really looking forward to working more with you and will always feel grateful to have known you.
à mes pots de labo, Djohan, Tifenn, Simone, Alessio et Paul. Venir tous les jours au travail a été amusant grâce à vous. Je sais que le laboratoire est entre de bonnes mains avec vous, je suis vraiment fier de vos récents résultats et le meilleur reste à venir.
Un grand merci à Julien, qui a repris mon projet de thèse et m'a guidé avec décision et m'a poussé à donner le meilleur de moi-même.
Merci à mes collaborateurs du CEA. L'équipe du pMUT (Emmanuel, Bruno et les autres) a vraiment soutenu mon premier gros projet résultant en une excellente publication. Inutile de dire que cela n'aurait pas été possible sans votre aide, merci beaucoup. Je suis heureux d'avoir eu l'opportunité de collaborer avec Aryballe (Pierre Maho notamment) et Marc Sansa, qui m'ont beaucoup appris sur les capteurs innovants.
Grazie ai miei amici di Grenoble. In questi tre anni siamo diventati una famiglia. Se mi sono legato cosí tanto a Grenoble, e sapete come sia cominciato il mio periodo qui, é merito vostro. Grazie ai miei amici di Casale. Mi siete mancati tantissimo, vi voglio troppo bene. Grazie a Sara, che ha reso la fine del mio percorso di dottorato più bella.
Grazie alla mia famiglia. Ai miei zii e cugini, a mia nonna, a mia sorella, a mia mamma e papá. Senza l'appoggio di mia mamma e papá non sarei sopravvisuto a quel primo mese in Francia e al periodo di confinamento. Mavi, in questi tre anni sei cresciuta parallelamente a me, anche se a distanza. Sono fierissimo dei tuoi traguardi e so che tu lo sei dei miei.

Abstract

Neuromorphic engineering is a field of research that leverages the latest technologies to build the next generation of computing and sensing systems inspired by biological neural systems. The term neuromorphic was coined in the late 1980s by Carver Mead, referring to electronic circuits closely matching the principles of biological computation. Today, neuromorphic research evolved into a multidisciplinary field gathering scientists from different fields and working towards different goals. In recent years, neuromorphic has gained popularity after the surge of deep learning with part of the computer science community drawing inspiration from biology for innovative computation. Neuromorphic engineering has also attracted many material scientists seeking applications for unconventional devices. The recent slowing down of Moore's law and the increasing costs of improving technological nodes have boosted the interest in unconventional computing strategies. Neural network algorithms, the backbone of artificial intelligence, are not conveniently implemented with Von Neumann computers, and In-memory-computing is emerging as a promising alternative architecture for artificial intelligence. This context motivates a paradigm shift for computing systems and the works in this thesis.
This thesis explores the field of in-memory computation with non-volatile-memories for neuromorphic systems. Resistive random access memories (RRAMs) are embedded into an analog architecture implementing a spiking neural network. Modern training techniques from deep learning unleash the full potential of analog in-memory computing. A modular architecture is proposed to scale the system to large-size graphical networks. RRAM-based computing is utilized to efficiently perform the localization task with auditory stimuli. A sensing system based on state-of-the-art piezoelectric-micromachined-ultrasound-transducers (pMUT) is paired to a bio-inspired spiking neural network, minimizing energy consumption. In-memory computing based on RRAMs is also applied to artificial olfaction, where an array of chemically functionalized Mach-Zender interferometers constitutes an innovative gas sensor. On-line learning with a dedicated circuit enables the adaptation of artificial olfaction at the edge. Non-volatile memories are also involved in novel forms of computations, going beyond the conventional schemes of neuromorphic computing. RRAMs endow synapses and neurons with plasticity mechanisms that coexist in an unsupervised learning procedure. Inspired by the intricate structures of biological neurons, dendritic circuits are proposed, extending existing network architectures constituted by

neurons and synapses. Dendrite circuits improve the efficiency and memory footprint of neuromorphic spiking neural networks. This thesis results are significant in the fields of in-memory computing and neuromorphic systems.

Résumé

L'ingénierie neuromorphique est un domaine de recherche dans lequel les dernières technologies sont utilisées pour construire la prochaine génération de systèmes informatiques et de détection, inspirés par les systèmes nerveux biologiques. Le terme neuromorphique a été inventé dans les années 1980 par Carver Mead, faisant référence aux circuits électroniques s'inspirant étroitement des principes du calcul biologique. Aujourd'hui, la recherche neuromorphique est devenue un champ multidisciplinaire regroupant des scientifiques de différents domaines et travaillant vers des objectifs différents. Ces dernières années, le neuromorphique a gagné en popularité après l'essor de l'apprentissage profond, avec une partie de la communauté informatique s'inspirant de la biologie pour du calcul innovant. L'ingénierie neuromorphique a également attiré de nombreux scientifiques des matériaux cherchant des applications pour des dispositifs non conventionnels. Le ralentissement récent de la loi de Moore et les coûts croissants pour améliorer les noeuds technologiques vont encore plus stimuler l'intérêt pour des stratégies de calcul non conventionnelles. Les algorithmes de réseaux de neurones, qui constituent la colonne vertébrale de l'intelligence artificielle, ne sont pas commodément implémentés avec les architecture de type Von Neumann, et le calcul en mémoire est en train de devenir une architecture alternative prometteuse pour l'intelligence artificielle. Ce contexte motive le changement de paradigme dans les systèmes informatiques et les travaux de cette thèse.
Cette thèse explore le domaine du calcul dans la mémoire avec des mémoires non volatiles pour les systèmes neuromorphiques. Des mémoire résistive de type RRAM (Resistive-Random-Access-Memories) sont intégrées dans une architecture analogique qui réalise un réseau de neurones à impolsions. Les techniques de apprentissage profond libèrent tout le potentiel du calcul analogique avec RRAMs. Une architecture modulaire est proposée pour étendre le système à des réseaux graphiques de grande taille. Le calcul basée sur les RRAMs est utilisé pour effectuer efficacement la tâche de localisation avec des signaux auditifs. Un système de détection basé sur des transducteurs piézoélectriques micro-usinés à ultrasons (pMUT) à la pointe de la technologie est couplé à un réseau de neurones à spike bio-inspiré, minimisant la consommation d'énergie. Le calcul en mémoire des RRAM est également appliqué à l'olfaction artificielle, où un réseau d'interféromètres Mach-Zender chimiquement fonctionnalisés implémente un capteur de gaz innovant. L'apprentissage en ligne avec un circuit dédié permet une nouvelle implementation de l'olfaction artificielle.

Les mémoires non volatiles sont également impliquées dans de nouvelles formes de calcul, dépassant les schémas conventionnels du neuromorphisme. Les RRAM dotent à la fois les synapses et les neurones de mécanismes de plasticité qui coexistent dans une procédure d'apprentissage non supervisée. Inspirés des structures complexes des neurones biologiques, des circuits dendritiques sont proposés, prolongeant les architectures de réseaux existantes constituées de neurones et de synapses. Les circuits dendritiques améliorent l'efficacité et l'empreinte mémoire des réseaux de neurones à spike neuromorphiques.
Les résultats de cette thèse sont significatifs dans les domaines du calcul en mémoire et des systèmes neuromorphiques.

List of Publications

Journal papers

[1] Filippo Moro et al. "Neuromorphic object localization using resistive memories and ultrasonic transducers". In: *Nature Communications* 13.3506 (2022), pp. 2041–1723. DOI: https://doi.org/10.48550/arXiv.2202.05094.

[2] Melika Payvand et al. "Self-organization of an inhomogeneous memristive hardware for sequence learning". In: *Nature Communications* (2022).

Conference papers

[3] Filippo Moro et al. "Hardware calibrated learning to compensate heterogeneity in analog RRAM-based Spiking Neural Networks". In: *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2022. DOI: https://doi.org/10.48550/arXiv.2202.05094.

Collaborations

[4] Thomas Dalgaty et al. "Hybrid neuromorphic circuits exploiting non-conventional properties of RRAM for massively parallel local plasticity mechanisms". In: *APL Materials* 7.8 (2019), p. 081125. DOI: 10.1063/1.5108663. eprint: https://doi.org/10.1063/1.5108663. URL: https://doi.org/10.1063/1.5108663.

[5] Yiğit Demirağ et al. "PCM-Trace: Scalable Synaptic Eligibility Traces with Resistivity Drift of Phase-Change Materials". In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401446.

Patents

[6] Thomas Dalgaty et al. “Synapse circuit for three-factor learning”. In: 17454435. 2022.

[7] Thomas Dalgaty et al. “Circuit and method for spike time dependent plasticity”. In: 17454373. 2022.

Contents

1 Introduction **11**
- 1.1 Neuromorphic engineering . . . 11
  - 1.1.1 Neuromorphic processors . . . 15
  - 1.1.2 Neuromorphic sensors . . . 18
- 1.2 Analog computing and memristors . . . 23
  - 1.2.1 Neuromorphic sub-threshold circuits . . . 25
  - 1.2.2 Memristors for Neuromorphic computing . . . 27
  - 1.2.3 In-memory-computing accelerators . . . 32
- 1.3 Learning in Neuromorphic Computing . . . 34
  - 1.3.1 Unsupervised Learning . . . 35
  - 1.3.2 Supervised Learning . . . 36
- 1.4 Scope of this thesis . . . 39

2 Building a Spiking Neural Network with RRAM **41**
- 2.1 Motivation . . . 41
  - 2.1.1 The limitations of RRAM-based ANNs . . . 43
- 2.2 Variability in analog electronics and RRAMs . . . 45
  - 2.2.1 Heterogeneity in Neurons and Synapses . . . 46
  - 2.2.2 Variability in RRAMs as Synaptic weights . . . 48
- 2.3 Neuromorphic Hardware-Calibrated Off-Chip training . . . 49
  - 2.3.1 Off-Line learning on SNNs . . . 49
  - 2.3.2 Energy Assessment . . . 57
- 2.4 Design of a RRAM-based SNN . . . 58
  - 2.4.1 RRAM Array . . . 59
  - 2.4.2 Array Periphery . . . 60
  - 2.4.3 Assembling the RSNN chip . . . 63
- 2.5 Scaling up with the Mosaic concept . . . 67
  - 2.5.1 The opportunities of small-world graphs computation . . . 67
  - 2.5.2 The Mosaic architecture . . . 69
  - 2.5.3 Application to real-time sensory processing . . . 75

3 Neuromorphic system for object Localization 79
3.1 Motivation . . . 79
3.1.1 Biological background . . . 81
3.2 PMUT sensors for Time-of-Flight measurement. . . . 83
3.3 RRAM-based neuromorphic computational map. . . . 87
3.3.1 Variability in neuromorphic circuits and RRAM-based calibration procedure. . . . 92
3.3.2 Mitigating RRAM variability with a dedicated programming strategy 92
3.3.3 A calibration procedure to minimize the impact of variability at the system level . . . 95
3.4 System assessment . . . 97

4 On-Line learning and Artificial Olfaction 102
4.1 Motivation . . . 102
4.2 Aryballe's Gas sensors . . . 105
4.2.1 Mach-Zender interferometer . . . 107
4.2.2 Collecting a dataset . . . 108
4.3 The Delta Rule for RRAMs . . . 109
4.3.1 From the algorithm to the circuit . . . 110
4.3.2 Computer-in-the-Loop experiments . . . 114
4.4 Transfer Learning with Delta Rule . . . 118

5 A step beyond in neuromorphic computing 125
5.1 MEMSORN: a memristive unsupervised self-organized map . . . 125
5.1.1 Motivation . . . 127
5.1.2 Endowing both Synapses and Neurons with plasticity: hardware implementation . . . 128
5.1.3 Technologically-plausible unsupervised learning . . . 132
5.1.4 Analysis on the effect of variability in SOSN . . . 135
5.1.5 Energy and latency estimations . . . 139
5.2 Dendritic Computation . . . 140
5.2.1 The Dendritic circuit element and the Dendritic Network . . . 141
5.2.2 Assessing the computational power of Dendritic Networks . . . 143
5.2.3 Hardware implementation of the Dendritic Network . . . 147
5.2.4 Extensions of the project . . . 151

6 Discussion 153
6.1 Perspectives on future work . . . 155

7 Appendices 159

Chapter 1

Introduction

1.1 Neuromorphic engineering

Neuromorphic engineering [8, 9, 10], or also neuromorphic computing, is a relatively young field of research in which advanced technology is used taking inspiration from, mimicking or seeking to understand biological neural architectures. Modern neuromorphic engineering is a broad, multi-disciplinary field unifying scientists with different backgrounds and with different scopes. The term "neuromorphic" has evolved through time and now embraces many novel approaches for unconventional, bio-inspired sensing and computing. To gain perspective on what neuromorphic engineering is and to understand where it is evolving, one has to look for the historical backgrounds. Starting from the precursor of neuromorphic computing - bio-inspired engineering - passing through the late 1980s and 1990s when the term was coined, and reaching the modern days where neuromorphic is one of the most popular research topics, with the promise of entering the market with competitive products.

History of neuromorphic engineering Inspiration from the models of neural computation and information processing taking place in the brain dates back to the foundation of computer science [11], in the 1950s. The spiking behavior of the nervous system inspired the abstract concept of point-neuron, postulated by McCulloch [12] in 1943. In 1958 [13], Rosenblatt proposed the first neural network inspired by the connectivity between neurons observed in the brain. This led to the flourishing field of Artificial Intelligence (AI), bloomed in the 1960s and 1970s [14]. An early example of physical implementation of a nervous system's element is the electronic retina, by Fukushima, in the 1970[15]. It is worth mentioning that the 1970s was the period of the first AI winter, in which funding and interest towards bio-inspired computation decreased. With novel advancements on neural network architectures [16] and gradient-descent learning [17], AI and bio-inspired computation re-gained traction as a research topic in the 1980s and 1990s. However, the

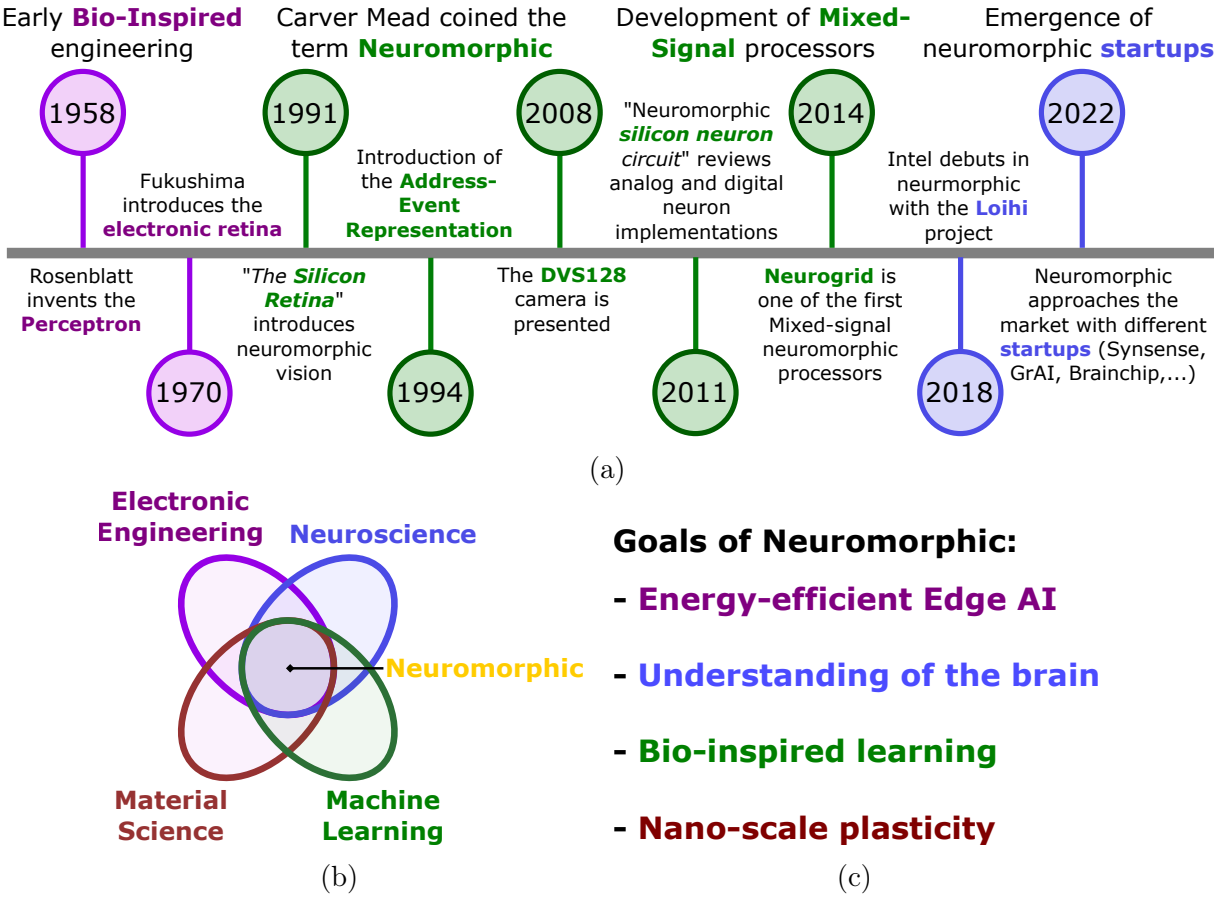


Figure 1.1: History and Definition of neuromorphic computing. (a) Timeline of the main events characterizing neuromorphic engineering, from the early days of bio-inspired engineering, through the development of the field under Carver Mead in the 1990s, up to the modern days with neuromorphic startups entering the market. (b) Neuromorphic can be defined as the field between 4 major disciplines, such as Electronic Engineering, Neuroscience, Machine Learning and Material Science. (c) As the neuromorphic field is highly heterogeneous, the objectives are as well. Among the main goals of neuromorphic the main ones are: building energy-efficient AI hardware, understanding the brain in a bottom-up manner, perform bio-inspired learning in machines and accomplish nano-scale plasticity.

utilization of very-large-scale-integration (VLSI) technology for bio-inspired computation and the emphasis on the non-linear and dynamical properties of biological computation only began in the late 1980s. A group of leading scientists involving Richard Feynman, John Hopfield and Carver Mead started the course "Physics of Computation", initiating the interest in unconventional computational methods. It was Carver Mead, especially fascinated by the biological aspects of computation and in collaboration with biologist Max Delbrück, who coined the term "Neuromorphic". Mead also contributed to some of

the first implementations of neuromorphic vision sensors (silicon retina [18] in 1991) and neuromorphic audition sensors (silicon cochlea [19]). It was in these years that the idea of building analog electronic circuit emulating the dynamics of biological neural elements was born, culminating with the Nature paper [20] presenting a "silicon neuron". With this legacy, neuromorphic engineering developed in the 1990s and early 2000s around Mixed-Signal processors and sensors. The creation of neuromorphic chips was accelerated by the invention of the Address-Event-Representation (AER) [21], efficiently communicating spikes across large areas. Among the most notable advancements of this period of neuromorphic research are the Dynamic-Vision-Sensor-128 [22] - an event-based camera featuring 128x128 pixels -, the Differential-Pair-Integrator neuron [23] and Neurogrid [24] - a large scale mixed-signal neuromorphic chip. Neuromorphic engineering is not bound to analog electronics, in fact digital versions of spiking-neural-networks (SNN) accelerators have gained traction in the last years. The ease in designing and producing such application-specific-integrated-circuits (ASICs) guided by neuromorphic principles, led to a new phase in which neuromorphic computing became promising as a profitable computing paradigm. In 2018, a major actor in the semi-conductor industry such as Intel launched the Loihi project [25]. Multiple neuromorphic startups (Synsense, GrAI, Brainchip, ...) populate the market today, mainly proposing digital ASICs. However, neuromorphic engineering is still mainly a research topic in universities, reaching many research groups across the entire world. Figure 1.1a sums up the main stages of the history of neuromorphic engineering.

Neuromorphic today The term neuromorphic has evolved particularly in this last decade, after the rise of Deep Learning [26]. More and more people from the Computer Science community have looked into biology to incorporate improvements in deep neural networks [27]. At the same time, engineers implemented these AI models mainly with digital electronic chips. Also, the advent of novel nano-scale electronic devices offered new hardware solutions onto which developing bio-inspired computation. All these influences make neuromorphic a highly heterogeneous field, that can be identified at the interface between 4 main disciplines (Fig. 1.1b): Electronic Engineering, Neuroscience, Machine Learning and Material Science. Giacomo Indiveri - one of the leading scientists in neuromorphic - claims in [9] that neuromorphic computing "aims to reproduce as faithfully as possible the detailed biophysics of the nervous system" and that neuromorphic systems make use of "spike for representing and processing signals". While this is certainly adherent to the spirit of the founders of neuromorphic, it has lately become usual to loosen the definition of the term and include a wider range of low-power computing solutions vaguely inspired by biology.
The mission of neuromorphic engineering is also not unique. In general, neuromorphic engineering aims at developing systems inspired by the neuro-physiology of the brain. This is differentiated in as many versions as the communities that populate the neuromorphic field. The result is that at least four main objectives can be identified (Fig. 1.1c): Energy-efficient Edge AI system, bottom-up understanding of the brain, perform bio-inspired

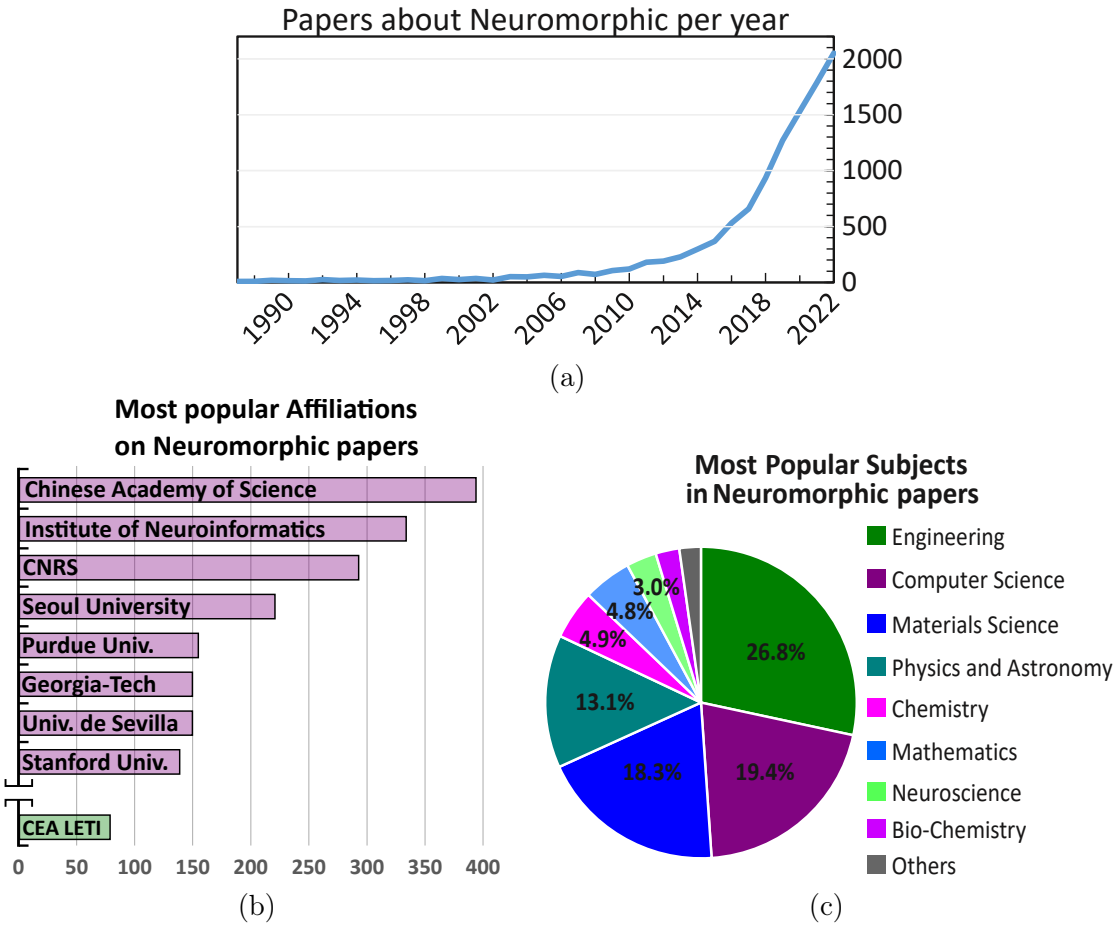


Figure 1.2: Neuromorphic as a research subject. Data collected from Scopus.com (a) Papers published about neuromorphic per year, from 1984 to 2022. The interest is exponentially increasing over the years, especially in the last 8 years. (b) Affiliations with the most publications in the field of neuromorphic. CEA Leti is a relevant research center in neuromorphic. (c) Subject related to the papers published on neuromorphic. The field truly multidisciplinary, with the largest contributions from (electronical) Engineering, Computer Science and Material Science.

learning in machines and novel devices capable of miniaturizing plasticity and learning. Neuromorphic engineering today is a very popular research topic, with exponentially increasing number of published paper every year, as demonstrated in Fig. 1.2a. Fig. 1.2b shows the most popular affiliations to papers published about neuromorphic engineering. CEA Leti is well positioned in the landscape of the research topic contributing with more that 80 papers. Fig. 1.2c reveals the multi-disciplinary nature of neuromorphic research, with most papers being about (electronic) engineering, computer science and material science. These statistics are available at "Scopus.com" [28].

1.1.1 Neuromorphic processors

At the heart of neuromorphic computing are neuromorphic processors. To build VLSI circuits tightly inspired by the neuro-physiology and faithfully reproducing the complex non-linear dynamics observed in the brain was one of the main goal of the founders of the subject. Different groups have come up with several solutions to build neuromorphic processors. In the first wave of neuromorphic, processors were mainly based on the Mixed-Signal paradigm, where analog neurons and synapse circuits are put in communications by AER or other digital communication protocols. More recently, fully digital implementations of spiking neural networks have gained traction and are nowadays commercially available. The main example of neuromorphic processors are presented and briefly in Table 1.1.

Mixed-Signal Closer to the spirit that initiated neuromorphic computing, mixed-signal processors utilize analog circuits that mimic the behavior of biological neurons and synapses. Analog circuits generally feature capacitor-based integrators, which emulate the charge integration of the soma, while synapses are conventionally implemented by voltage or current-gated circuits, modulating the charge transfer between neurons. While computation is perfomed in the analog domain, spikes are generally routed with digital communication protocols, among which the most common is the Address-Event representation (AER). In AER, spikes are assigned an address which represents the post-synaptic neuron. The address allows the spike-packet to pass from core to core and finally be delivered to the target neuron. One of the earliest and most representative mixed-signal neuromorphic processors is **Dynap-SE** [29]. Developed in the Institute of Neurinformatics (ETH and UZH) under the supervision of Prof. Indiveri, the processor is the basic version in a family of other similarly built chips (Dynap-SE2, Dynap-CNN, Dynaps-SEL). The processor features 4 cores with 256 neurons in each core, and up to 64k synaptic connections. Communication between neurons is mediated by AER protocol, divided in 3 routers. The chip can be interfaced by USB and can take AER spike packets coming from other event-based sensor or chips. **Braindrop**[30] is one of the very few Mixed-Signal chip fabricated with an advanced technology node (28 nm). Its simplified Integrate-and-Fire neurons and optimized AER routing scheme minimize energy consumption, reaching an impressive 388 fJ per synaptic operation (energy per hop). The architecture is also one of the most compact, at 0.65 mm^2. **BrainScaleS-2** [31] is a processor fitting in the Human-Brain-project [32], and it has been developed at the Heidelberg University. The processor is built in a modular design where different chips are placed and put in communication forming a big cluster. The analog neurons and synapses in BrainScaleS are designed in the analog domain utilizing 65 nm technology and work in accelerated time (1000x respect to real-time). This means that the time constant of a neuron in BrainScaleS is 3 orders of magnitude faster than that of the biological counterpart it emulates.

Digital Closer to conventional computers in the physics of computation, but running neuromorphic algorithms, digital neuromorphic processors simulate spiking neural networks. Neurons and synapses are built with different degrees of complexity and are free of the mismatch issue, typical of analog circuits. As they don't require capacitors to emulate the membrane potential behavior, neurons and synapses generally occupy much smaller sizes than their analog counterparts, so digital chips tend to scale to larger network sizes. Digital neuromorphic processors can benefit from the highly developed infrastructures of Computer-Aided-Design (CAD) tools to efficiently design large scale chips. This is why digital architectures are the preferred choice for big electronic manufacturers approaching neuromorphic, or startups envisioning new innovative products. One of the first digital chips is **TrueNorth**[33] from IBM, 2014. The chip features 1 M neurons packed in 4.3 cm^2 (full chip size). Neurons are arranged in 4096 cores forming a 2D mesh, made to optimize the communication at the network level. The physical neuron block in TrueNorth uses time-division multiplexing to compute the states of 256 logical neurons for a core with a single computational circuit. TrueNorth was built with a modular architecture, where several chips can work together. As demonstrated in [33] 4 TrueNorth chips can perform an object detection task on a High-Definition video in real time, consuming less than a Watt. **Loihi** [25] is the neuromorphic processor from Intel, presented in 2018. The chip is based on a 14 nm FinFET process, comprehending over 2 billion transistors and 33 MB of memory, in the size of 60 mm^2. The neurons are grouped in 128 cores, composing a 2D mesh. Neurons can be programmed to simulate different neuron types, from simple Integrate-and-Fire to complex multi-compartment models. Loihi exploits time-multiplexing to compute the state of multiple neurons within the same neuron circuit. Also, the chip contains three x86 processing units devoted to routing and controlling the communication between the cores. Loihi also features on-line learning capabilities with simple learning rules such as Spiking-Timing-Dependent-Plasticity.
These chips constitute the first generation of neuromorphic processors. The aim of the chips is to explore the power of event-based computation, offering versatility and accessibility to users in research. This is why they are not tailored to a specific task and thus not competitive to highly optimized architectures, such as ASIC accelerators for Artificial Intelligence. Nonetheless, they had a key role in popularizing neuromorphic computing to people outside electronic engineering, providing easy access to biologically inspired computational models.

With the field of Edge-AI blooming in the last decade, more and more ASICs have been built, some of them running spiking neural networks. Most of these architectures are not strictly adherent to the initial spirit of neuromorphic computing, but are rather guided by the optimization of figures of merit such as performance and energy efficiency. Some others made the choice of incorporating elements of neuromorphism, such as spiking neurons and event-driven computation. Emulating Leaky-Integrate-and-Fire neurons and running Spiking-Neural-Networks, **ODIN** [34] and **ReckON** [35] are two chips implemented with digital electronics, simulating task-agnostic recurrent network architectures. ODIN features

	DYNAPSe[29]	TrueNorth[33]	Braindrop[30]	BrainScaleS 2[31]	Loihi[25]
Producer	INI (ETH/UZH)	IBM	Stanford	Heidelberg University	Intel
Year	2017	2014	2018	2022	2018
Process Technology	Mixed-Signal 180nm	Digital (ASIC) 24nm	Mixed-Signal 28nm	Mixed-Signals 65nm	Digital (ASIC) 14nm
Die Size	43.79 mm^2	4.3 cm^2 (chip size)	0.65 mm^2	-	60 mm^2
Number of Neurons	1024	1M	4096	512 per core	130k
Number of Synapses	64k	265M	16M	130k	130M
Total Memory	64kB CAM + 4kB SRAM	12.75kB per core 4096 cores	-	-	256MB SRAM
Energy per hop	17pJ @ Vdd = 1.3	2.3pJ @ Vdd= 0.77	381fJ	-	15pJ

Table 1.1: Comparison of the main neuromorphic processors. Mixed-Signal processors such as DynapSE, Braindrop and BrainScaleS are closer to the original neuromorphic spirit. Truenorth and Loihi are digital architectures achieving greater density of integration, featuring larger numbers of neurons and synapses. The low-power nature of these processors is expressed by the energy for transmitting a spike (energy per hop) which is in the 1-10 pJ range for most architectures.

learning in the form of Spike-Time-Dependent-Plasticity and ReckON features on-line learning capabilities with the E-Prop learning rule (more on that in Section 1.3). Both chips are implemented in 28 nm technology and present high energy efficiency per synaptic operation: 8.4-14.2 pJ for ODIN and 0.6-42 pJ for ReckOn. ODIN was demonstrated on image recognition, ReckON on key-word-spotting, image recognition and navigation.
NullHop [36] is a FPGA-based implementation of an accelerator based on conventional artificial-neural-networks, but enhanced by the neuromorphic concept of temporal and spatial sparsity in the architecture. The concept runs a heavily optimized convolutional neural network performing computer vision tasks, where activations of hidden layers are sparsified. The concept is tested on a 28 nm technology emulator yielding 450 GOp/s operating a VGG19 network at 500 kHz. The neuromorphic concept of sparsity also inspired Spartus [37], an accelerator for recurrent neural networks. The idea of Spartus is to communicate activations between neurons just when a large enough change occurs. This minimizes communication and optimizes energy efficiency. The Spartus concept reaches 1.1 TOp/s/W on a FPGA implementation.
In general, a lot of work has been devoted in the last few years to lower the power budget of Edge AI accelerators. Neurormorphic not only paved the way for efficient computation

with the processors presented in Table 1.1, but also inspired other efficient implementations of digital accelerators.

1.1.2 Neuromorphic sensors

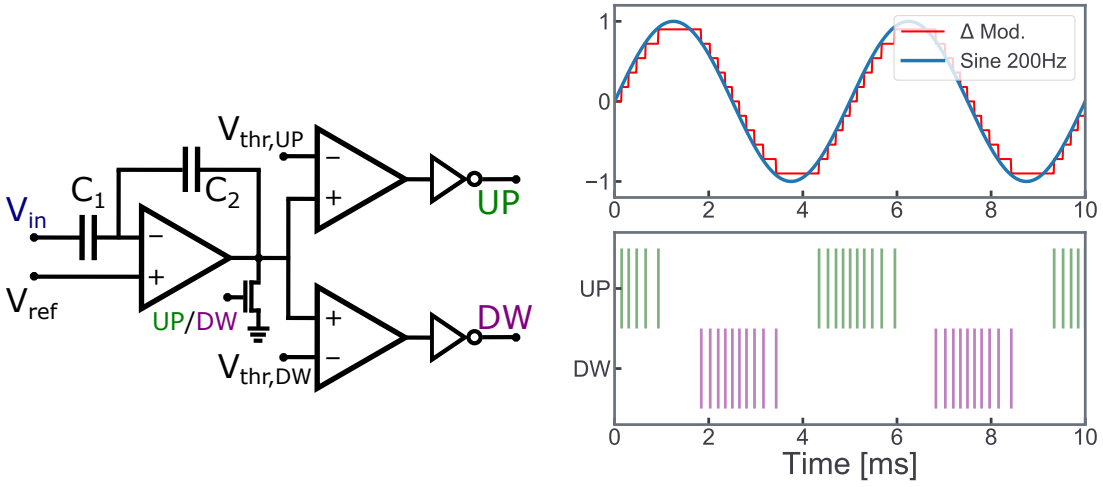
As important as neuromorphic computing is neuromorphic sensing. In analogy to the nervous system, while a neuromorphic processor would represent the brain, neuromorphic sensors emulate the biological senses. Sensors provide the stimuli and information computed by the processor, so it is fundamental that they work well together. From the beginning, neuromorphic engineers have focused on mimicking biological senses. Carver Mead, in the late 1980s and early 1990, put a lot of efforts to design circuits that would reproduce the retinal cells [18] and the cochlea [19]. Neuromorphic vision and audition were born. Starting at the circuit level and developing into full systems in the late 1990s and early 2000s, neuromorphic sensors are now wide-spread in research laboratories and have also entered the market. Neuromorphic sensors are designed around the core neuromorphic concept of being event-based. This means that sensors don't communicate signals while information is not detected, rather only when a stimulus is present. This is the distinguishing factor of neuromorphic sensing.

Converting information to the spike-domain

In neuromorphic sensors, analog input stimuli are encoded into spikes. The same happens in biology. This is necessary in the nervous systems of animals - particularly mammals - as sensory stimuli have to travel long distances to reach the central processor, the brain. Analog dynamical information would be lost by travelling across the nerves, while spikes can carry information more faithfully in the form of either frequency or precise timing. Receptors of biological sensors feature different mechanisms to translate information into electricity, and signals all converge in the brain in the form of action potentials, also said spike. Similarly, neuromorphic sensors all feature different technologies to capture input stimuli, but all translate it to the spike-domain. It is then evident that the conversion of analog information to spikes is the key aspect in common between neuromorphic sensors.
How to represent analog signals with spikes?
A number of different techniques has been developed to convert information to the spike domain, trying to optimize two key factors:

- Information retention
- Energy efficiency

The most basic technique to translate analog intensity to the spike domain is *Rate coding*. Intensity of a signal is matched to the frequency of spikes. Similar spike conversion mechanisms are observed for muscle activation and tactile sensing in biology [38]. This technique requires large latency to optimize information retention and it generally results in poor energy efficiency. A more effective spike encoding technique is *Temporal coding*



(a) (b)

Figure 1.3: Conversion of information to the spike domain. (a) Delta Modulator, simplified circuital implementation. (b) Output Gain profile as a function of the input audio frequency, from all the 64 filters in the DAS's filter back. From [40].

[39], where intensity in the analog domain is translated into latency in eliciting a spike. The stronger the analog signal, the earlier the spike is elicited. This technique is efficient to encode static images into spikes, but less adapted for temporal signals.

The most popular circuit to convert information from the analog to the spike domain is the Delta Modulator. Delta Modulation is a well known analog-to-digital and digital-to-analog technique to translate analog signal by means of pulses. In the delta modulator, an input signal is processed by analysing its derivative and producing spikes only when the signal changes over time. For this reason, the technique is event-based. A simplified realization of the delta modulator circuit is shown in Fig. 1.3a. An input signal is presented at V_{in} and passes through C_1 to remove the DC component. An integrator circuit sums the increments of the input signal V_{in} on the capacitor C_2. When these increments overcome either the positive or negative thresholds $V_{thr,UP}, V_{thr,DW}$, the two output comparators emit an output spike at the UP or DW outputs. Figure 1.3b sums up the working principles of the circuit. V_{in} is represented in blue and the signal reconstructed from the spikes emitted by the delta modulator circuit, is in red. Green and Violet spikes are the output of the delta modulator circuit.
This circuit, in different forms and with different features is the basic building block in common with most neuromorphic sensors.

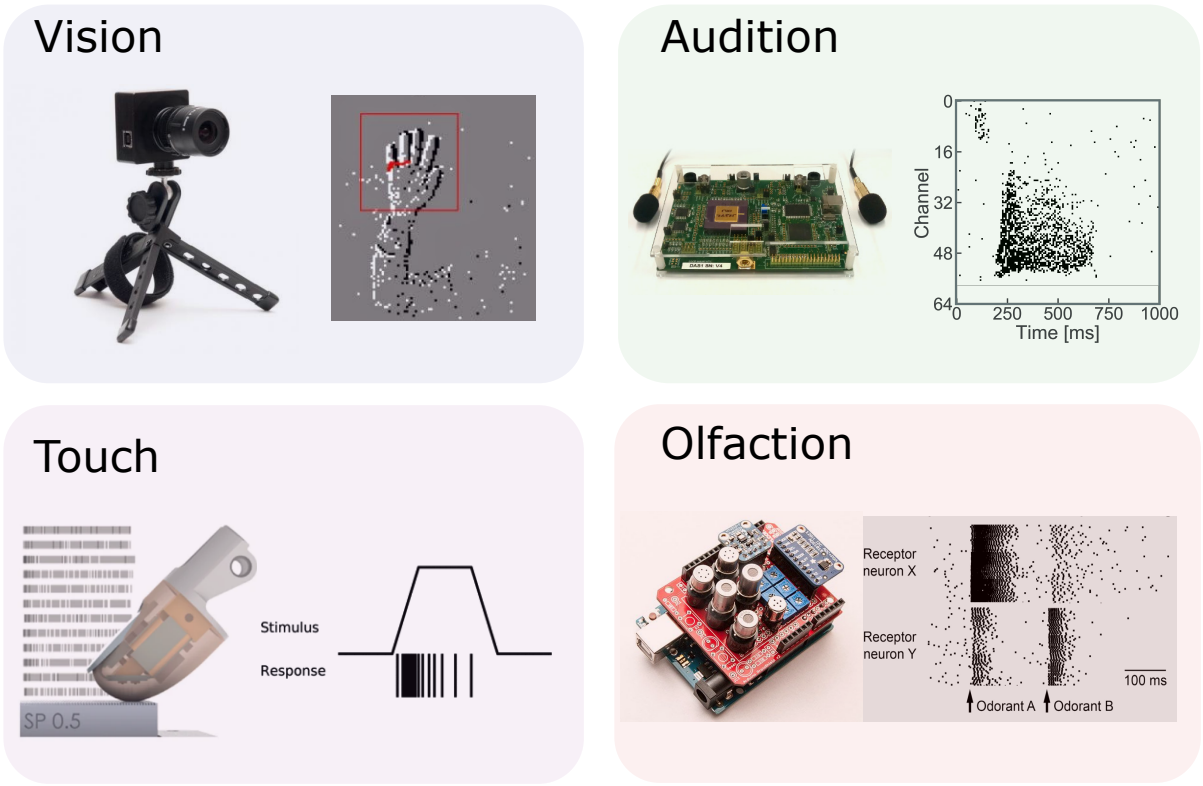


Figure 1.4: Neuromorphic sensors. (Vision) Event-based camera are commercially available and feature unmatched dynamic range, low latency and low power consumption. The inset on the left shows a DAVIS128, one of the first prototypes of event-based camera. On the right, a waving hand is recorded on a DAVIS128. Images from inivation.com. (Audition) Neuromorphic audition is inspired by the biological cochlea, differentiating acoustic stimuli based on their frequencies. A model of a stereo Dynamic Audio Sensor (DAS) is shown, taken from inivation.com. The response of the sensor is a train of spikes for each of the channel it features, where each channel encodes the intensity of a particular frequency of the input audio. In the inset on the right, the word "four" was spelled. (Touch) Tactile sensors can utilize different technologies, among which capacitive and optical readout. These sensor mimic the biological response of sensory cells in the skin, eliciting a train of spike under stimulation. Figure from [41]. (Olfaction) The concept of electronic nose is conjugated in the neuromorphic domain with an array of gas sensors producing trains of spike in response to the presence of a particular molecule. The gas-sensor in the inset shows a Metal-oxide gas sensor from Alpha MOS. The figure on the right shows the response of different receptor to different olfactory stimuli, observed in the Drosophila [42].

Neurorphic senses

While in the first phase of neuromorphic engineering research has focused around models of the retina and cochlea, neuromorphic sensing is nowadays divided in 4 blocks, covering the

main biological senses: Vision, Audition, Touch and Olfaction. Figure 1.4 shows the most prominent examples of neuromorphic sensors. Neuromorphic vision is mainly performed by event-based cameras (DVS128 [22] in the inset) producing spikes at each pixel when the luminosity changes over time. Audition generally exploits conventional microphones, whose output is translated to the spike domain and later processed. The figures shows a dynamic-audition-sensor [43] and the spiking output it produces when listening the word "three". Touch is a less investigated sense in neuromorphic. The figure shows a capacitive touch sensor built over a prosthetic finger [41]. To follow the behavior of biological tactile receptors, the sensor can be calibrated to respond at the onset of a stimulus and to decay its activation over time. Gas receptors are common sensors commercially available in the market. In the last years they received interests from the neuromorphic community and different solutions have been developed to convert olfactory stimuli to the spike domain. As shown in the figure, certain population of neurons can respond maximally to a specific input pattern registered by the gas sensor and poorly to others, thus helping in classifying olfactory stimuli.
Neuromorphic is extremely relevant for embodied intelligence and robotics [44, 45] too. To endow robots with artificial intelligence is perhaps one of the leading themes of the next phase of research in neuromorphic. In this sense, neuromorphic intelligence is more explainable and trustworthy than its artificial counterpart. Neuromorphic sensing, with it low-power and low-latency nature is a perfect fit in power- and latency-constrained use cases such as robotics.

Vision Vision is certainly the most developed field in neuromorphic engineering. From the early implementation of the silicon retina [18] in 1991, to the pioneer dynamic-vision-sensor [46, 22], event-based cameras are nowadays commercially available. Compared to a conventional camera, neuromorphic vision offers unmatched dynamic range (up to 120 dB), low latency (in the order of 10 μs) and low power consumption (down to the 10 mW range). The general operation of even-based camera is that of a conventional photo-receptor paired with a Delta-Modulator circuit. The logarithm of the produced photocurrent in a pixel is buffered and converted to the spike domain. This means that when the scene presents no changes in luminosity over time, the camera produces no output, according to the event-based nature of the pixels. Neuromorphic cameras communicate the stream of output events sensed by the pixels making use, in most cases, of a AER protocol. This allows them to be paired to a neuromorphic processor. State-of-the-art event based cameras are approaching the High-definition resolution and are implemented using advanced technology nodes. Table 1.2 shows the most advanced versions of event-based cameras, produced by IniVation, Prophesee, Samsung and CelePixel.

Audition Research in neuromorphic audition began when Lyon and Mead developed an auditory model inspired by the human cochlea [19]. Unlike a conventional signal processing model for audio, sampling sound at a certain rate, a silicon cochlea is formed by

Supplier	iniVation	Prophesee	Samsung	CelePixel
model	DAVIS346	Gen 4 CD[47]	DVS-Gen3[48]	CeleX-IV[49]
year	2017	2020	2020	2019
resolution	346 x 260	1280 x 720	1280 x 960	1280 x 800
dynamic range (dB)	120	>124	100	120
power consumption (mW)	50-175	32-84	130	400
chip size (mm2)	8x6	6.22x3.5	8.4x7.6	5.3x5.3
CMOS technology (nm)	180	90	65/28	65

Table 1.2: Comparison of commercially available event-based cameras.

a cascade of filters selective to a certain frequency band. The filters encode the intensity of input frequencies which are later transformed to the spike domain with a delta modulator circuit, just like what an inner-hair cell would do in a biological cochlea. By tackling the issue of mismatch and refining the design, modern cochlea implementation reach $>$ 60 dB of dynamic range, feature high quality factor overlapping filters and consume a fraction of a milli-Watt. The latest silicon cochlea communicate the output spikes with the AER protocol, ensuring a perfect match with neuromorphic processors. Combining the silicon cochlea circuit with Micro-Electric-Mechanical-System (MEMS) microphone, the EARAER [40] is one of the most advanced audition sensors. It features 32 output channels, biologically inspired filter cells, and ensure sparse output activation with Integrate-and-Fire neurons to translate the channel activation into spike. The sensor has been tested in a localization task [50]. Further improvements of this sensor led to the development of EARAER2 [43, 51], which comes with 64 output channel, improved tunability of the filter bank, greater tolerance to temperature changes and improved dynamic range.

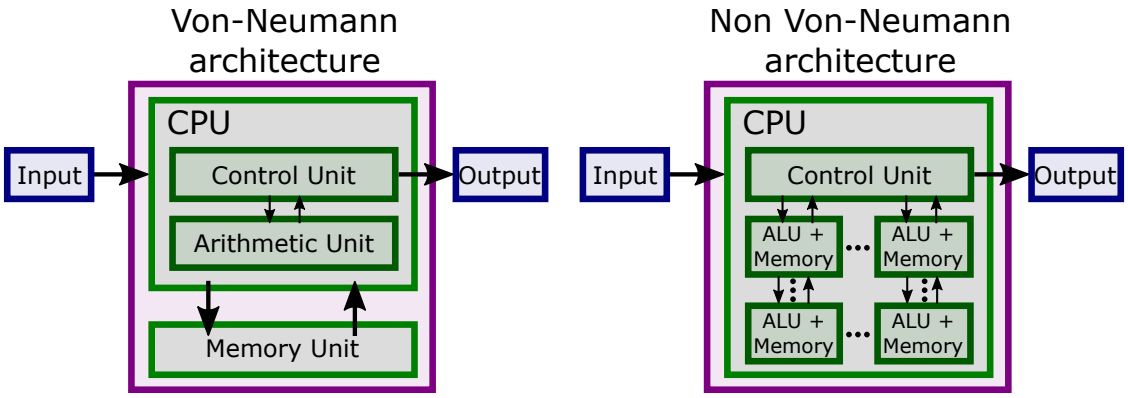
Touch The electronic skin is an innovative concept for artificial tactile sensors with applications in robotics [52] and prosthetics [41]. Leveraging state-of-the-art technology, the electronic skin aims at reproducing the precision and dynamic range of human touch. Biology guides the advancements in artificial touch. For example, tactile receptors are not uniformly distributed across the skin, so do in the latest electronic versions [53]. Humans possess different types of tactile receptors, each with different sizes and shapes, responding to different stimuli: it would be desirable that electronic skin integrates different technologies to enhance touch precision and dynamic range. The most commonly adopted sensing modes are capacitive, resistive, piezoelectric [54], optical and magnetic. Current sensors mainly measure pressure, altough it would be desirable to also capture stress, shear and vibrations. A key property to perform touch sensing in robots and prosthesis is that the substrate onto which the sensors are distributed have to be flexible and resistant to stress and pressure, as mentioned in [55]. Exploiting the available technology, neuromorphic touch sensors could improve robots performance in object sensing and manipulation, as well as enhancing the sense of touch in humans.

Olfaction The development of electronic sensors for measuring odors began in the 1960s [56] and the concept of electronic nose was introduced in the the 1980s [57]. From that point on, research focused on portability and low power solution for artificial olfaction. The most popular sensors in electronic noses are Metal-Oxide-Sensors (MOX) [58], Conductive Polymers, Quartz-Crystal Mircobalance (QCM) and Acoustic-Wave Sensors. Lately, optical sensors have been proven as compact and low-latency solutions for odor sensing [59]. Neuromorphic engineering has contributed to artificial olfaction mainly from the algorithm side. A conventional modern electronic nose features arrays of gas sensors with low selectivity and extracts a *signature* from this high dimensional measurement with machine learning algorithms. While most electronic nose implementations do not include a stage of conversion of information to the spike domain, biologically inspired models of odor classification and learning rules have been proposed [60].

1.2 Analog computing and memristors

Computers are programmable machines that can perform logic or arithmetic operations. Despite only being popularized in the second half of the twentieth century, early example of computers have been built thousands of years ago. They were very different from what computers look like now, and mainly exploited mechanical principles. Examples of prehistoric computers are the abacus used to ease the counting, the Antikythera mechanism believed to be the first analog computer and capable to compute the astronomical position of stars, the tide-prediction machine built in the late nineteenth century by William Thomson. One of the first modern electronical computers was built during the second world war: ENIAC [61] (electronic numerical integrator and computer), acted as a catalyst in developing electronic systems by convincing many about the potential of numerical computation. At the same time, in 1946, the transistor was invented: this electronic device transformed the computing industry, driving the development of digital computers from the 1960s up to these days. The improvements of the performance and size of transistors - called transistor scaling - allowed computer to pack more and more components while consuming less power. Transistor scaling was studied by Gordon Moore, which stated the famous "Moore's law" [62], claiming that - among other predictions about the performance of computers - *the number of transistors on a microchip doubles every two years*. The rule proved true throughout the whole development of Very-large-scale-integrated (VLSI) technology, and is only lately showing a departure from the prediction. Technology has reached an impressive rate of innovation, having evolved multiple times to maintain momentum in improving performance. From the early planar bipolar technology, transistors have then evolved to the complementary-metal-oxide-semiconductor (CMOS) technology, taking the shape of fully-depleted-silicon-on-insulator (FD-SOI), FinFETs and lately Gate-All-Around [63] transistors. Transistor size has shrunk considerably and now reached critical dimensions. While consumer electronics runs 7 and 5 nm processes [64], IBM claimed a new technological node called 2 nm in 2021 [65]. Despite the impressive scaling of transistor size, technology is struggling to keep up with the Moore's Law, particularly

concerning the advancement in frequency of operation and power consumption [66]. A recent article from the IEDM conference highlights the limitation in Static-Random-Access-Memories size and performance improvements in the latest technological nodes [67].



(a) (b)

Figure 1.5: Central-Processing-Unit Von Neumann simplified architecture with its memory bottleneck. (a) Simplified schematics of the architecture of a CPU. Inputs interact with a control unit which activates the arithmetic and logical unit to perform the required computation. To perform such computation, memory data have to be fetched from the memory unit, constituting the memory Von Neumann bottleneck. (b) By embedding together the memory and arithmetic, the Von Neumann bottleneck is overcome. In general, in Non-Von-Neumann architecture, computation and memory is co-located and distributed. This architecture is often referred to as In-Memory-Computing (IMC).

.

Computer performances are getting harder and harder to improve and some say they will eventually saturate with the current technology. On top of this technological limitation, conventional digital processors have an architectural constraint: the Von-Neumann bottleneck. Most of the computers in the market follow the Von-Neumann architecture (Fig. 1.5a), thus fetching data from memory blocks which is then used for computation. This data movement is the main source of energy consumption in modern processors [68, 69]. This limitation is particularly severe when running deep learning models, constituted by millions or even billions of parameters needed to be fetched from memory. To overcome this limitation, a new class of architectures has been proposed, under the umbrella-term Non-Von-Neumann architectures. The general scheme of such design distributes both computation (Arithmetic-Logic-Units, ALU) and memory and integrates them together (Fig. 1.5b). The result is that such inherently parallel processing scheme does not impose data movement in favour of a more distributed and local computation. When computation and memory are truly co-located, one refers to such architecture as In-Memory-Computing (IMC).
Innovating forms of computation not only concern architectural paradigm, but also

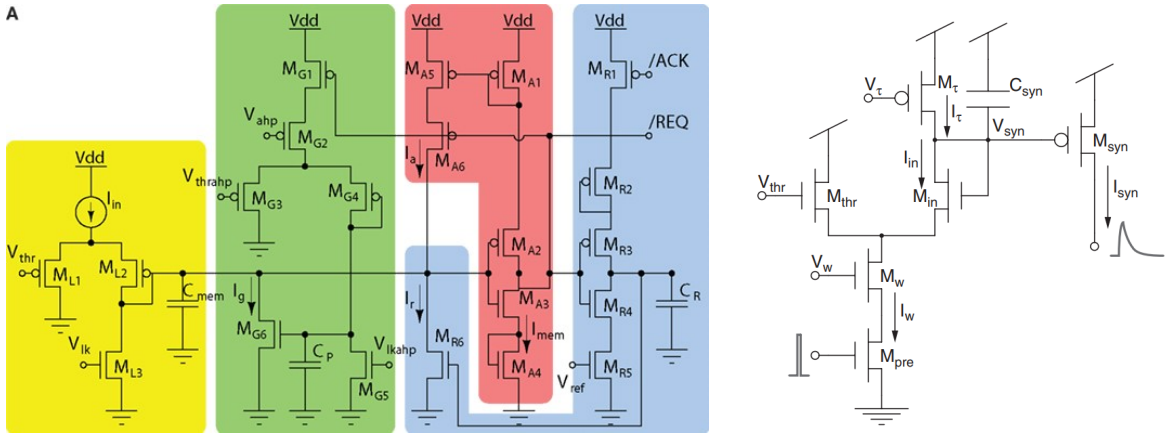
re-purpose the role of transistor. Analog computing exploits the highly non-linear sub-threshold region of transistor and makes full use of the analog representation of values, rather than their digitized versions in conventional computers. Analog electronic circuits naturally implement complex dynamical systems evolving with according to differential equations, requiring high computational power and precision when computed in the digital domain. In neuromorphic computing, the properties of analog circuits are leveraged to reproduce the complex dynamics of neural systems.
This context of slowed down technological scaling and the relevance of the Von Neumann bottleneck for artificial intelligence applications form the perfect conditions for a paradigm shift in computation. Neuromorphic is a possible solution to improve in particular areas in computation, such as artificial intelligence at the edge. However, neuromorphic and non-Von-Neumann architectures are not to replace conventional computers for general purpose use-cases. They are meant to be an improvement in the specific cases where data movement causes severe performance limitations.
Neuromorphic computing requires new technological and architectural substrate that exploit the parallel and non-linear nature of biological computation. Analog In-Memory Computing is a promising candidate to become the paradigm of choice for the next generation of neuromorphic processors. This computational paradigm joins the sub-threshold circuits mimicking neural elements together with novel devices with exotic properties.

1.2.1 Neuromorphic sub-threshold circuits

The shift of paradigm in computation does not only involve an architectural change, but also a technological one. Since transistor scaling has slowed down, the natural consequence is to look for more performance out of the single transistor, going beyond the digital regime. This is also what Carver Mead advocated when founding neuromorphic computing. By making use of the sub-threshold regime, the high non-linearity of the transistor can be exploited for computation. Following this principle, Mead and his team developed the silicon neuron and synapse [20] in 1991. Over the course of the following two decades, several circuit implementations of neurons and synapses have been proposed. Today, the most popular sub-threshold neuromorphic circuits are based on the differential-pair-integrator (DPI). The DPI is a two-transistor circuit that scales an input current and drives it towards a capacitor, where it is integrated. This circuit has become popular as it constitutes the sweet-spot between complexity of circuit and fidelity with respect to a biological neuron.

DPI Neuron and Synapse The DPI neuron [23] is perhaps the most relevant silicon neuron circuit and it implements a generalized Leaky-Integrate-and-Fire model. In its most complete form, the DPI neuron is composed of 4 blocks which implement various functions: Integrating the input current from synapses, Generating Spikes Event with aid by a positive feedback, Resetting the membrane voltage to the resting potential after a spike and Adaptation to the past activity. With reference to Fig.1.6a, this blocks are

respectively colored in Yellow, Red, Blue and Green. Simpler versions of the circuit might only feature the Yellow and Red blocks.



(a) (b)

Figure 1.6: Neuromorphic sub-threshold CMOS circuits. (a) Differential pair integrator neuron circuit, from [23]. The circuit is composed of 4 main blocks, highlighted with different colors. The yellow block the is the DPI, taking the input current I_{in} and charging it over the capacitor. The red block generates an output when the threshold voltage is overcome. The refractory period is generated by the blue block and adaptation by the green one. (b) Differential Pair Integrator Synapse circuit, from [70]. An input voltage pulse is scaled by M_W and integrated on C_{syn}. The circuit outputs a synaptic current I_{syn}.

From a qualitative point of view, the behavior of the circuit can be described as follows, as from [23]. Input currents I_{in} are modulated by the differential-pair circuit (Yellow), as a function of the bias voltage V_{thr}. The modulated input charges the membrane voltage capacitor C_{mem}, which is in parallel to the leakage transistor ML_3. This transistor determines the time-constant of the neuron, by controlling its gate with the bias V_{lk}. As V_{mem} approaches the switching voltage of the inverting amplifier $M_{A2} - M_{A3}$ (Output section, Red), the feedback current I_a starts to flow through $M_{A5} - M_{A6}$, increasing V_{mem} more sharply. The output spike triggers the refractory period circuit (Blue), in which M_{R6} maintains V_{mem} to ground, as long as the voltage across the capacitor C_R is high. During the spike emission period (occurring when terminal REQ exceeds the voltage threshold), a current with amplitude set by V_{ahp} is sourced into the adaptation section (Green) of the DPI neuron, with a gain set by the gate bias voltage $V_{thr,ahp}$, and a time constant set by $V_{ik,ahp}$. The adaptation current I_g increases with every spike, following the same first-order dynamics of I_{mem}. Following the approach of the Translinear Principle, one can extract the equation regulating the whole circuit in its sub-threshold dynamics. Before the positive feedback is activated, generating the spikes, the circuit behaves according to

the following equation.

$$\tau \frac{dI_{mem}}{dt} = -I_{mem}\left(1 + \frac{I_g}{I_\tau}\right) + I_{mem,\infty} + f(I_{mem}) \tag{1.1}$$

$$\tau_g \frac{dI_g}{dt} = -I_g + I_{g,max} r(t) \tag{1.2}$$

In this equation, the non linear term $f(I_{mem}) = \frac{I_a}{I_\tau}(I_{mem} + I_{th})$ depends on both the membrane current and the positive feedback current. The two time-constants τ, τ_g and the currents $I_\tau, I_{\tau,g}, I_{mem,\infty}, I_{g,max}$ are defined accordingly:

$$\tau = \frac{CV_t}{\kappa I_\tau} \qquad \tau_g = \frac{CV_t}{\kappa I_{\tau,g}}$$

$$I_{\tau,g} = I_0 e^{\frac{\kappa V_{ik}}{V_t}} \qquad I_\tau = I_0 e^{\frac{\kappa V_{ik}}{V_{ikaph}}}$$

$$I_{mem\infty} = \frac{I_{in}}{I_\tau} I_O e^{\frac{\kappa}{U_T} V_{thr}} \; I_{gmax} = \frac{I_{MG2}}{I_{\tau g}} I_O e^{\frac{\kappa}{U_T} V_{thradp}}$$

The DPI synapse (Fig. 1.6b) follows similar dynamics, as it shares the same differential-pair-integrator input circuit. Inputs arrive in the form of voltage pulses at the transistor M_{pre}. They are weighted by the transistor M_W, biased with a voltage representing the synaptic weight (V_W). Voltage bias V_{thr} modulates the current generated by M_W and M_{pre}, which is then integrated onto C_{syn}. The transistor M_τ sets the time constant of the circuit, by the leakage current I_τ. The output is the first order filtered and weighted response of the input pulse and it converted to the current domain by means of the transistor M_{syn}.

Such circuits are the basis of most of the works involving sub-threshold neuromorphic circuits utilized in this thesis. They promise to lower the cost of computation in biologically inspired machine learning, in the context of spiking neural networks.

1.2.2 Memristors for Neuromorphic computing

While sub-threshold circuits constitute non-linear and dynamical computational elements, memristor can play the major role of memory element in In-Memory-Computing, going beyond Von Neumann architectures. Memristors - theoretically conceived by Chua in 1971 [71] - are electronic devices modulating charge transfer across them, whose conductivity depends on its past states. Memristors have been experimentally demonstrated in 2008 by HP [72] and have ever since been fabricated utilizing a number of different designs, material, exploiting several physical effects. For beyond Von Neumann architectures, memristors are particularly interesting as memory elements that can be integrated with logical units. Material scientists have developed different memristors that can be integrated with CMOS technology and enable analog In-Memory-Computing architectures [73, 74].

A particular interesting class of memristor are Non-Volatile-Memories (NVM). These are two or three terminal devices with programmable conductance, retaining information in their conductive state without the need of stand-by power consumption. By contrast, the static-random-access-memories (SRAMs) and dynamic-random-access-memories (DRAMs) require a voltage to maintain information stored in the form of a digital bit. Instead, Flash Memory - based on floating gate transistor technology - is non-volatile, but it requires high voltage for programming and presents limited endurance. Innovative Non-Volatile-Memories offer more compact, CMOS compatible and better performing memory devices that could replace conventional memory units.

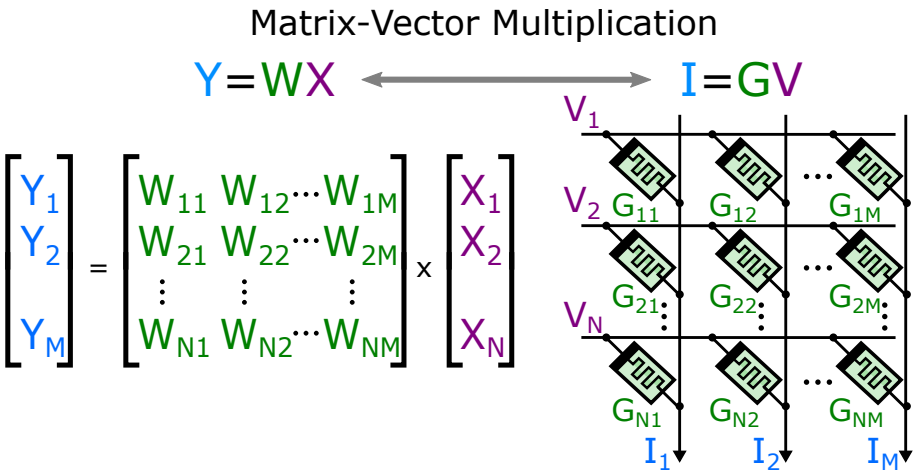


Figure 1.7: In-Memory-Computing with memristive crossbars. The crossbar efficiently performs the Matrix-Vector-Multiplication (left), ubiquitous in neural networks. The operation involves an input (X) that is encoded as an array of voltages (V) set as the potential at the rows of the crossbar. The matrix term (W) is translated into the conductance (G) of the memristors in the crossbar array. The output (Y) results from the current at the columns of the crossbar (I).

NVMs for In-Memory-Computing are particularly suited for neural network implementations. In artificial intelligence, the most common operation is multiply-and-accumulate (MAC), involved in matrix-vector-multiplication (MVM). In conventional Von-Neumann machines, this operation requires fetching the inputs (input vector and matrix elements) from memory, performing the multiplication and partial summations. Exploiting the benefits of In-Memory-Computing with NVMs this operation is conveniently carried out by the Ohm's law on a crossbar of memristive devices. Summation is replaced by the union of two currents converging at the same wire ($I_{sum} = I_i + I_j$), multiplication is performed by applying a voltage to a NVM with a certain conductance G: $I_{mul} = V \times G$. The crossbar architecture is depicted in Fig. 1.7. The input vector (X) is input to the crossbar's rows as an array of voltages (V). The matrix (W) is encoded by the conductance of the devices (G). The output of the MVM operation (Y) is the resulting current at the columns of the crossbar (I). Performing this operation in-memory results in much higher energy efficiency

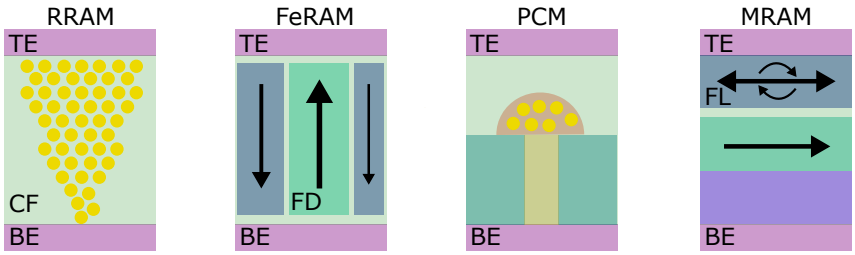
when running machine learning models.

Non-Volatile-Memory types From the inception of memristors in 2008 [72], material scientists have developed numerous devices exploiting different materials and physical effects. The four main classes of non-volatile-memories are depicted in Fig. 1.8a: resistive-random-access-memories (RRAMs), ferroelectric-random-access-memories (FeRAMs), phase-change-materials memories (PCMs), and magnetic-random-access-memories.

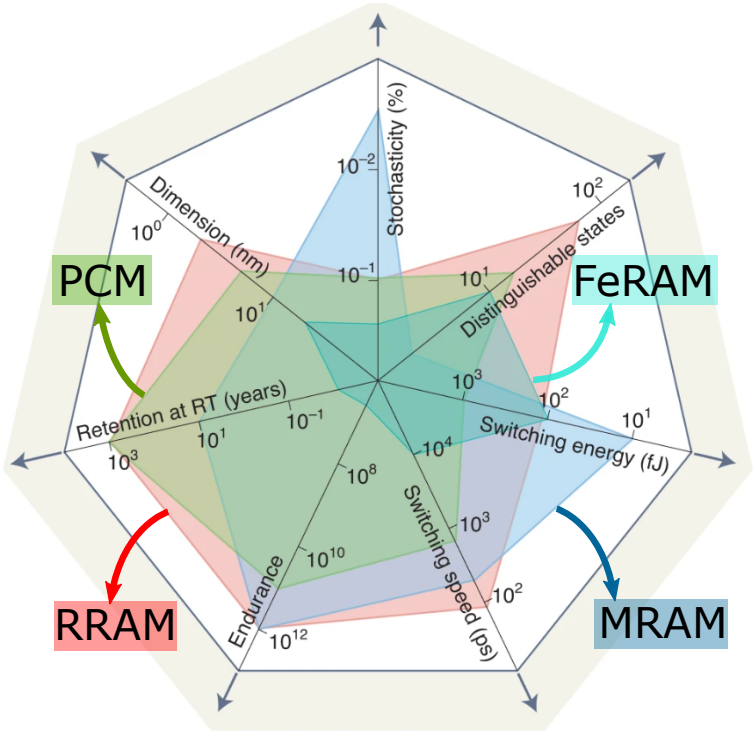
RRAM devices store information in a conductive filament that is formed across an insulating material. The filament can be controlled with SET and RESET operations, equivalent of WRITE and ERASE operations in conventional digital memory cells. One of the most common process integration stack to build RRAMS is $Ti/TiN/HfO_2/Ti$, in which a hafnium-dioxide insulating layer is sandwiched between 2 titanium electrodes. Thanks to its ease of integration, RRAMs offers low production cost compared to other NVMs. RRAMs also have scaled down in size down to 10 nm in lateral dimension [76], making them interesting for advanced technological nodes as well. The nanoscopic scale of the conductive filament and its imperfect stability make RRAMs conductance imprecise both when the filament is formed and when its disrupted.

FeRAM [77] is a recently developed class of NVMs exploiting the ferroelectric effect to store information. SET and RESET operations change the state of the device as the ferroelectric domain in the ferroelectric material sandwiched between two electrodes. Similarly to DRAMs, reading FeRAM device is a disruptive process, meaning that the information contained by the device is removed. FeRAM offer much higher data retention (10 years at 85 K) and read/write endurance (up to $10^{10} cycles$) than Flash device [78]. The most common material used to fabricated FeRAM is lead-zirconate (PZT), even though recently Si-doped HfO_2 has also been demonstrated exhibiting ferroelectric properties. The advantages of FeRAMs are their low-energy, high endurance, fast write operation and its resistance to magnetic fields. Notably, ferroelectric materials have also been used as insulators in transistors, forming the ferroelectric-field-effect-transistor (FeFET) [79] and the ferroelectric-tunnel-junction (FTJ) [80] device. The FeFET is a three terminal device with tunable threshold voltage, depending on the orientation of the ferroelectric domains in the insulator. The FTJ is similar to a FeRAM in construction, but it features an additional non-ferroelectric insulator layer which allows current to pass exploiting the tunneling phenomena only. This yields low conductance and thus extremely low current (and power) when reading the device state.

PCM devices are based on phase-change materials. The most commonly used material in PCMs is the GST ($Ge_2Sb_2Te_5$). This class of materials, also said chalcogenide, can be in a crystalline or amorphous phase. These two states of the material are characterized by different conductivity which result in two main conductive states of the device, representing the stored information. The crystalline phase is a high conductance state (HCS) and the amorphous phase corresponds to a low conductance state (LCS). The switching



(a)



(b)

Figure 1.8: Non-Volatile-Memories (a) Illustration of the stack of the 4 main NVM types: RRAM, FeRAM, PCM and MRAM. They respectively exploit the Redox, ferroelectric, Phase Change and Magnetoresistance effects to encode and store information in the device. (b) Comparison of performance between the 4 mentioned NVMs, from [75].

mechanisms between the two states requires a current flowing between the electrodes heating the chalcogenide by the Joule phenomena. As a drowback, PCM suffer from a structural relaxation mechanism that makes the conductance of the devices drift in time when in the amorphous state [81].

MRAM devices utilize the magnetic domains to store information. The integration

stack of MRAMs is rather complex. In its basic configuration, two metal electrodes sandwich a device composed by multiple layers. An Antiferromagnetic layer (A) at the bottom supports a pinned ferromagnetic layer which never changes its magnetic domain. On top of that, an insulating layer is deposited. Finally a second ferromagnetic layer, called Free Layer (FL), is the one responsible for storing information in the cell. When its magnetic domain is parallel to the pinned layer, conductance in the device is high due to the tunneling magneto-restistance phenomenon. When the orientation of the magnetic domain in the free layer is anti-parallel to the pinned layer, the conductance is low. The switch in magnetization in MRAM happens via the programming current that induces a magnetic field, capable of switching the free layer magnetization. This leads to high current densities during programming which have limited the scalability of MRAM arrays. A possible solution is the introduction of spin-torque-transfer devices, in which a polarized current is used to turn the spin of the free layer and thus change its magnetization.

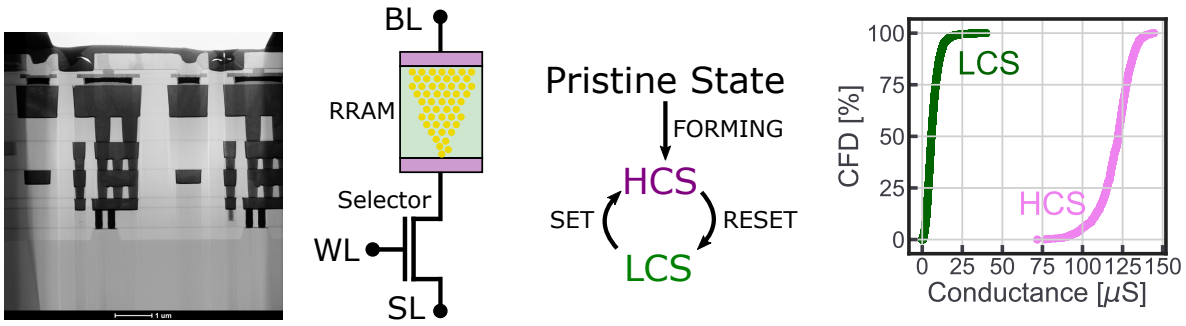
Figure 1.8b compares the mentioned NVM technologies in 7 main performance metrics. RRAMs and PCMs are the ones with the highest potential in scaling the dimension of the bit cells and also provide high number of distinguishable conductance levels. FeRAMs and MRAMs lead in switching energy, which is down to 100fJ. All the technologies offer great endurance ($> 10^9$ cycles) and retention at room temperature (> 10 years). This shows the great potential of Non-volatile-Memories as memory devices.
Thanks to the performance of NVMs, they have been utilized in commercial products. Intel and Micron released in 2015 the 3DXpoint memory, based on PCM devices, resulting in the Intel *Optane* [82] product. Panasonic released microcontrollers utilizing RRAMs as read-only-memories [83]. ST-Microelectronics unveiled a microcontroller for automotive applications featuring a 16 Mb PCM array in 2018 [84]. Samsung started the commercialization of MRAMs embedded on a 28 nm FD-SOI node in 2019 [85].

RRAM characterization As described before, RRAM devices are two terminal NVMs featuring Top Electrode (TE) and Bottom Electrode (BE) sandwiching an insulator material. Via redox reaction, a filament is formed under high enough electric field across the insulator. The conductive filament can be disrupted by with an electric field of opposite polarity. SET and RESET operations are performed to write and erase the device, thus forming and disrupting the filament. After a SET operation, the information is stored as a high conductance value and RESET brings the device back to a low conductance state. RRAMs exist in many configurations of different materials for both electrodes and storage structure. The electrodes can adopt metals such as Cu, Ag, Pt and Au [86], carbon nanotubes [87] and nitrides such as TiN [88]. The insulator is commonly built as a binary oxide containing HfOx [89], MgOx [90], SiOx [91] or AlOx [92]. At CEA-Leti, RRAMs are mainly based on HfO_2 oxides, with thickness between 5 and 20 nm. This type of stack reports an ON/OFF ratio of up to 10^5, nano-second scale read operation and endurance cycles from 10^6 to 10^9 [93]. This technology can achieve up to 10 years of data retention at 125^oC. RRAM devices are paired with an access transistor, which allows to

individually select the cell for programming and reading. Paired to an access transistor, RRAMs form the 1T1R device. Figure 1.9a shows a scanning-electrons-microscope figure of a 1T1R device fabricated at CEA Leti in the Back-End-of-Line (BEOL) of a 130 nm CMOS process. With this type of technology, devices are fabricated with crystalline HfO_2 insulating layer. At this stage, the insulator presents low defects density and no conductive filament, thus being in the so-called Pristine State. In order to form a conductive filament a first one-off programming operation has to be performed: the Forming operation. When forming is successful, a filament has been generated and the RRAM enter its normal programming loop where SETs and RESETs are alternated to write and erase the memory cell (Fig. 1.9b). High and low conductive states (HCS and LCS) are characterized by variability, inherent to the intrinsic random atomic arrangement of the conductive filament. The result is that LCS and HCS measured from 1024 RRAM devices programmed all with the same conditions present a distribution of conductance values (Fig. 1.9c). Variability is both at the device level and at the population level. In the first case, one refers to the cycle-to-cycle (C2C) variability. The same device is programmed alternating SETs and RESETs in the same conditions while the device conductance is recorded. Due to the stochasticity of the filament, distributions similar to those in Fig. 1.9c appear. An analogous affect is obtained when programming a population of devices with the same programming condition. In that case one refers to device-to-device (D2D) variability. To overcome variability, particular programming techniques have been developed. Enhanced programming strategies are particularly useful to extract more bits of information out of the RRAM cell. While the LCS is weakly modifiable, the HCS state can be controlled with the current imposed during the SET operation, the compliance current. It has been shown in [94] that RRAMs can reach up to 32 distinguishable levels. Exploiting the whole conductance range is particularly desirable for artificial intelligence applications, in which RRAMs are involved in neural network computations. The more distinguishable levels and the more precision in conductance, the higher the accuracy of the model. Later in this thesis, the effect of variability of RRAMs in a neural network will be analysed in details.

1.2.3 In-memory-computing accelerators

Recent advancement in non-volatile-memory technologies for In-Memory-Computing have pushed different research groups and industry divisions to implement their IMC Edge AI accelerator. The common ground in these implementation is the crossbar architecture, where NVMs are organized in a array performing the matrix-vector-multiplication. A team in CEA-Leti built **Spirit** [95], a IMC RRAM-based accelerator implementing a spiking-neural-network. This chip features Integrate-and-Fire neurons and classifies the MNIST dataset with 88% accuracy. Thanks to the non-volatility and low power nature of SNNs, the chip - built on a 130 nm process - consumes 180 pJ per synaptic operation. However, the team estimates that the energy per hop can be lowered at 17 pJ with a more advanced technological node (28 nm). RRAMs are at the heart of another impressive Edge AI accelerator from the Tsinghua University and University of Massachusetts [94].



(a) (b) (c)

Figure 1.9: Non-Volatile-Memories (a) Scanning Electron Microscopy image of a RRAM device built in the Back-End-of-Line of a 130 nm CMOS process. (b) 1TR configuration of the RRAM device, with the NVM device connecting to the bit line and the access transistor (selector) connected to the word line and source line. The device is initially in the Pristine state, and can be initialized or "formed", forming the conductive filament. High-Conductive-State is when the filament is formed and is obtained through a SET operation, Low-Conductive-State is when the filament is disrupted and is obtained with the RESET operation. (c) SET and RESET characteristic yielding the HCS and LCS distribution. This plot shows the large memory window of RRAMs programmed in a binary fashion.

The chip is built with a 130 nm process and can emulate both a multi-layer-perceptron classifying handwritten digits at 98% accuracy or convolutional-neural-networks (CNN) classifying the CIFAR-10 dataset with 88% accuracy. Thanks to the clever mapping of the CNN on to a 2,048 element array of RRAM, the chip performs 11 TOP/W (tera-operations-per-Watt). IBM efforts in PCM-based in-memory computing culminated with **Hermes** [96]. The chip is built with 14 nm, one of the most advanced nodes ever integrated with NVMs, and performs 10.3 TOP/W. Notably, the chip features current-controlled-oscillators as sense-amplifiers for the PCM devices, taking advantage of their analog conductance levels. Samsung presented a chip in 2022 [97] featuring MRAMs employed for IMC. This implementation is innovative as it solves one of the main limitations of MRAM: their high conductance range, leading to large currents. By performing the MAC operation by summing the resistances instead of the currents, the chip drastically lowers the currents along the rows of the devices and thus the power consumption. The chip reaches a record-breaking 405 TOP/W with a binary-neural-network performing image classification. **NeuRRAM** [98] is a prototype chip presented by a research group lead by Gert Cauwenberghs at the University of California, San Diego. The chip features RRAM devices and is built with a 130 nm technology node. It is one of the first large-scale implementation of IMC, involving 48 cores each with a 64 kbit RRAM array. Key to the performance of the chip is a clever communication protocol between the cores and the clever layout of neurons and programming circuits for the RRAMs array..

	Spirit [95]	[94]	Hermes [96]	[97]	NeuRRAM [98]
Producer	CEA Leti	Tsinghua University	IBM	Samsung	UCSD/Pittsburg/ Tsinghua University
Year	2019	2020	2021	2022	2022
Memory technology	RRAM	RRAM	PCM	MRAM	RRAM
Process node	130nm	130nm	14nm	28nm	130nm
TOP/W	/	11	10.3 (8bits ops)	405 (1bit ops)	/

Table 1.3: In-Memory-Computing accelerators for Edge Artificial Intelligence applications. IMC architectures have employed RRAMs, PCMs and MRAMs, showing competitive power efficiency, evaluated as Tera-operation per Watt (TOP/W).

1.3 Learning in Neuromorphic Computing

Learning is the process of acquiring understanding, knowledge, skills and behavior. From the point of view of neuromorphic computing, learning is adapting to new environments and inputs, understanding and solving new tasks. While in the theory of computer science many learning algorithms have been developed, here we focus on the ones that are - to some extent - inspired by the brain or that are relevant for neuromorphic computing. Back-propagation for gradient descent - the bread and butter of machine and deep learning, is biologically implausible and computationally intensive: the algorithm requires to store to memory the activation of each neurons, the weight updates are non-local and, in most cases, the algorithm involves the parallel computation of multiple input data points (batches). On top of these hardware-related issues, gradient descent applied to deep neural network generally requires huge amount of input data to take advantage of the size of the network. It follows that despite reaching unprecedented and impressive results in artificial intelligence, the back-propagation algorithm is very inefficient both at utilizing hardware resources and in generalizing with few data points. It is well known that the state-of-the-art deep learning models are trained on clusters of graphical-processing-units and consume enormous amounts of energy to get trained. For example, GPT3 [99] - one of the most powerful language models, with up to 170 B parameters - was trained on a cluster of V100 NVIDIA GPUs - taking approximately 27 years of GPU time - for an estimated total cost of >4.6 M dollars. The problem is not only economical: the CO_2 released to train GPT3 is calculated at around 35'000 kg, the equivalent to that emitted by 5 households in a year [100]. Modern deep learning based on back-propagation requires more and more computational power, energy and money to keep growing. On top of that, deep learning models are also a lot less efficient that humans to learn. In computer vision, the state-of-the-art image classification models have for long been trained on the Imagenet [101] dataset. 14 millions images are presented to the deep learning models multiple times, over the course of several training epochs. Despite this data-intensive method, computer vision models only recently have approached human-level abilities at classifying the Imagenet dataset [102, 103], while humans only required a handful of sample

images to abstract the concepts of - say - *cat*, *dog* or *car*. These issues call for a shift of paradigm in learning algorithms and the human brain is an example where learning is extremely efficiently with respect to the the low power budget and to the little data it receives. Neuromorphic computing aims at joining forces with computational neuroscience and machine learning to find out more about learning and plasticity in the brain and to transfer this knowledge into neuromorphic processors.
Following the conventions of machine learning, learning algorithms can be grouped into 3 main categories: unsupervised, supervised and reinforcement learning. Unsupervised learning algorithms make use of unlabelled input data and group them into clusters. Supervised learning exploits labeled data to learn the mapping function between the input and its labels. Reinforcement learning involves an agent in an environment whose aim is to explore and maximise reward for taking specific actions. The following is a brief review of the main neuromorphic approaches to the learning dilemma. Neuromorphic learning mainly aims at satisfying two main issues: the spatial and temporal credit assignment. The issues deal with how to modify each synapse in a network (from the input layers to the output) and when to perform such update, still being compatible with on-line learning requirements (batch size of 1), retain biological plausibility (locality of weight update) and aware of the computational substrate, i.e. neuromorphic processors. Neuromorphic reinforcement learning is not explored in this section as not an important part in the rest of the manuscript.

1.3.1 Unsupervised Learning

The motivation for developing biologically plausible unsupervised learning is the assumption that most of the stimuli presented to biological systems are non-labelled, yet mammals - and humans in particular - still exhibit astounding intelligent behaviors. In computational neuroscience, unsupervised learning was founded by Hebb in 1949 [104]. Following the so-called hebbian principle "*neurons that fire together, wire together*", many unsupervised learning rules have been proposed: spike-timing-dependent-plasticity (STDP), Oja's rule, spike-drive synaptic plasticity (SDSP) and others. Most of them are based on the correlation between two synaptic layers, as stated by the hebbian principle. The hebbian principle has been observed in in-vitro experiments [105] and is non-local. For this reason, hebbian learning is compatible with neuromorphic processors and does not require high computational loads. **STDP** is the most popular form of unsupervised learning and it is based on the precise timing of the post-synaptic neuron respect to its pre-synaptic neuron. If the post-synaptic neuron's activity is positively correlated with its afferent, the weight between the two is to be incremented. Otherwise, the weight is depressed. In its basic formulation, the weight change exponentially decreases with the spike time difference between pre- and post-neuron. The formula expressing such learning rule is then:

$$\Delta w_{ij}(s) = \begin{cases} A_+ exp[s/\tau_+] & \text{if } s < 0 \\ A_- exp[-s/\tau_-] & \text{if } s > 0 \end{cases} \tag{1.3}$$

where w_{ij} is the synaptic weight between the pre (j) and post (i) neurons. $\tau_{\mp}$ are the time constant of the positive and negative correlation between spike timing.
While STDP is still the main form of unsupervised learning, it suffers from some well known issues such as weight-saturation and stability. **Oja's rule** was introduced to solve these problems, adding little complexity tot STDP. The formula of the Oja's rule can be expressed as:

$$\Delta w_{ij} = \mu a_i (a_j - a_i w_{ij}) \tag{1.4}$$

where $a_{i,j}$ is the activity or firing rate of the pre and post neuron respectively. Despite its superiority respect to STDP, the added complexity made the introduction of the Oja's rule limited in neuromorphic computing.
Spike-Dependent Synaptic Plasticity (**SDSP**) is a modification of the STDP learning rule, where at the post-synaptic spike, a weight update is computed based on the activity of the pre-synaptic neuron. Membrane potential and spike frequency are common proxies of the activity of the pre-synaptic neuron in SNNs. A two threshold system determines whether to perform a positive or negative weight update, or whether not to perform one. The learning rule can be summarized in the following equation:

$$\Delta w_{ij}(s) = \begin{cases} -\mu & \text{if } a_i < \Theta_- \\ +\mu & \text{if } a_i > \Theta_+ \\ 0 & \text{if } \Theta_- < a_i < \Theta_+ \end{cases} \tag{1.5}$$

where a_i is the activation of the pre-synaptic neuron, $\Theta_{\mp}$ the positive and negative threshold of activation to enable the learning rule. SDSP, thanks to its hardware friendly learning rule, has been widely implemented in neuromorphic hardware [106, 107].
Hebbian learning rules have been dominant in the first phase of neuromorphic learning but have lately decreased in popularity. In fact, hebbian learning is not compatible with deep networks, and especially fails in the spatial credit assignment problem. For this reason, it is hopeless to train a deep network - either a spiking or artificial one - with unsupervised learning solely. After all, despite in small amounts, humans receive labelled data in the form of textbooks, spoken lessons and annotated figures from a young age. For neuromorphic learning to be successful, unsupervised plasticity has to be supported by supervised learning.

1.3.2 Supervised Learning

The domain in learning that saw the greater development in neuromorphic computing is certainly supervised learning. Gradient descent based algorithms are the ones achieving the best performance and their popularization over the course of last decade, the deep learning era, involved neuromorphic computing as well. However, neuromorphic networks, SNNs, present a problem: they are non-differentiable models. The activation function of neuron is a Heaviside function, which has 0 derivative except in the threshold of the activation, where the derivative is not defined. Such activation function blocks the gradients and the

disables the back-propagation algorithm. To work around the problem, different techniques have been proposed: [108] first train an equivalent differentiable ANN and then convert it to SNN; [109] worked on the exact spike timing and derived an analytical model for leaky-integrate-and-fire neurons so to compute a gradient based on the membrane potential; [110] also developed a differentiable model based on the membrane voltage of neurons allowing an approximated gradient to flow. All these technique had severe shortcomings: [108] was limited to work in the rate coding regime, eliminating the temporal sparsity of SNNs; [109] method only allowed neurons to spike once during training and imposed high computational complexity to back-propagate the gradient; [110] also required heavy computational loads to perform the back-propagation. It wasn't until 2019 that SNNs found a good implementation of the back-propagation algorithm.

Surrogate Gradient technique With the Surrogate Gradient technique [111], spiking neural network could finally take on deep learning models. The simple idea enabling back-propagation in SNNs is to replace the zero-derivative activation function of neurons with a surrogate function, only used in the back-ward pass. This function allows a fictitious gradient to flow across the layers of the SNNs, despite the zero-derivative activation function of neurons. It has been thoroughly demonstrated that this technique can reach performance at the same level of artificial neural networks [112]. More on this technique will be described in Chapter 2. While this technique is certainly not biologically plausible - as it is an extension of the back-propagation algorithm - it has been fundamental in the development of neuromorphic computing in the last 3 years. Thanks to this learning scheme, neuromorphic processors have reached unprecedented performance and have been proven competitive to Edge-AI accelerators [113]. However, the surrogate gradient technique is not the end-goal of neuromorphic learning and it presents several shortcomings: first, it i not biologically plausible; second, it requires high computational loads; third, just like conventional gradient descent techniques, it requires a lot of data and a lot of iteration to learn effectively.

Avoiding Back-Propagation To overcome the limitations mentioned above, computational neuro-scientists and researches in computer science have come up with alternative algorithms that avoid the back-propagation of gradients in neural network, still performing gradient descent. An early version of such techniques is Real Time Recurrent Learning, or **RTRL** [114] which is an algorithm computing the gradients across a network during the forward pass. The algorithm has only been rarely used as it requires $O(n^4)$ computational complexity, where n is the number of neurons. Other methods are three-factor-learning-rules. They are called that way as they involve a pre-synaptic, a post-synaptic and a global term in the weight update. An particular version of such algorithms has gained tremendous popularity in the neuromorphic community: **e-prop** [115]. E-prop is a biologically plausible learning rule compatible with on-line learning and with neuromorphic processors. With a computational complexity of $O(n^2)$, e-prop makes use of eligibility

traces, as synaptic storage units of the activity of neurons. Making use of the eligibility trace, post-synaptic activation and a global error signal, e-prop has been demonstrated to approximate the back-propagation algorithm, thus yielding great performance. E-prop is also compatible with reinforcement learning. Different solutions have been proposed to implement E-prop with neuromorphic hardware [35]. The limitation of e-prop is mainly that it is not optimized for deep networks. While technically applicable to any architecture, it introduces more and more approximations with increasing number of layers, thus lowering its performance compared to back-propagation.
In the family of energy-based algorithms, Equilibrium propagation [116] or **Eq-prop** is an emerging learning rule in neuromorphic computing. Eq-prop is based on a re-formalization of neural networks into dynamical systems where weights are bi-directional. The learning scheme is divided in two phases: forward and nudging. In the forward pass, the network converges into an equilibrium state. In the nudging phase, the correct label nudges the corresponding output neuron by a certain amount, causing a perturbation in the network. The network finds another equilibrium state. Based on the two equilibria, the weights are updated with local information only. This learning rule approximates back-propagation and has been demonstrated on complex datasets [117, 118].
Very recently, Hinton - one of the godfathers of deep learning - proposed the **Forward-Forward** algorithm [119]. The algorithm avoids back-propagation and proposes two separate forward passes. In the first pass, the algorithm makes use of the input data and the weight updates aim at improving the "goodness" of any hidden layer. The second forward pass is based on "negative" input data and aims at decreasing the goodness of the hidden layers. The algorithm is in its first steps of evolution and has already been demonstrated to efficiently solve modestly challenging computer vision tasks. Hinton claims that the algorithm is particularly suited for a novel class of computers in which software and hardware are co-designed.

Meta-Learning and Meta-Plasticity Still in the category of supervised learning, meta-learning focuses on exploiting meta-data to improve upon conventional learning procedures. Well established meta-learning procedures are MAML [120] and Reptile [121]. They are based on the idea to use meta-tasks to meta-train a neural network. After the meta-training, the network is in a state in the parameter space where it is easier to learn new tasks. This endows the network with few-shot learning abilities, closer to that of humans, thus enhancing the efficiency of learning compared to standard supervised methods. The method also has important hardware positive implications as it promises to lower the precision requirements for the weight updates when learning new tasks [111]. Memristive based neural network might benefit from such learning scheme.
Another form of enhanced learning is meta-plasticity. Meta-plasticity involves different learning rules with the aim of including the history of synaptic efficacy in the weight update, so to improve memory formation in neural networks. A popular implementation of this concept is reported in [122], where the authors implemented meta-plasticity in binarized neural network to mitigate the issue of catastrophic forgetting. The authors

claim that with a bio-inspired meta-plastic learning rule, a neural network can learn multiple tasks incrementally, without forgetting the previously learned ones.

Self-Supervised Learning Self-supervised learning is the paradigm that boosted deep learning in its latest phase of development. Taking supervised learning a step forward, self-supervised learning does not require labelled data, it rather exploits the features in unlabelled data to acquire knowledge. This enabled to scale up the size of datasets and not require human supervision for producing labels. In self-supervised learning, after a first phase of pre-training on unlabelled data, the model is generally fine-tuned over a small set of labelled data. In deep learning, self supervised learning has been used in state-of-the-art vision and language models. Only recently, such learning paradigm has been explored for neuromorphic models [123]. This paradigm is potentially fundamental to enable a new level of performance in neuromorphic model. Neuromorphic datasets are scarce and labelled data even rarer. Self-supervised learning will hopefully promote the creation of large scale unlabelled neuromorphic datasets, which are more easily generated, and will boost the performance of neuromorphic computing.

1.4 Scope of this thesis

The scope of the thesis project is to contribute to the field of neuromorphic engineering proposing novel systems integrating computing and sensing for applications in Edge AI. The common ground to the projects explored throughout the thesis is the exploitation of the Non-Volatile-Memories developed by CEA Leti. The focus is on RRAM devices due to their characteristics and process maturity. Resistive memories are employed in event-driven in-memory computing architectures, taking advantage of the non-volatility of the devices and the temporal sparsity of spiking-neural-networks. Interest towards biological solutions to perform efficient computation guides most of the projects in this thesis, with the idea that neuromorphic systems are a potentially disruptive technology.
Given that neuromorphic is a highly multi-disciplinary field, this thesis spans over different topics and investigates aspects that belong to integrated circuit design, device characterization, and novel algorithms development. Projects in this thesis created new interactions with experts in material science, to follow the latest developments in integrated devices, biologists, from which to gain knowledge about biological systems, electronic designers, to help transform the acquired knowledge into electronic circuits.
The thesis is organized in four chapters addressing independent projects. However, the projects share the same intention to develop biologically inspired electronic systems centered around RRAMs. Altogether, they provide a proposition for the next generation of Edge AI systems, shifting computational paradigm from digital Von-Neumann architectures to analog In-Memory computing. The benefits but also the limitations of the proposed systems are analysed and addressed, strengthening the propositions.
The first chapter is dedicated to the development of a RRAM-based spiking neural network circuit. Exploiting the latest advancements in learning algorithms, the problematic of

variability in RRAMs and analog circuits is mitigated. A RRAM array is used as the basis for a memristive SNN and the concept of a modular design is proposed to increase the scale of SNN hardware implementations. Similar circuits are employed in an object localization task - in the second chapter - performed by combining advanced miniaturized ultrasound sensors and a neural network topology inspired by that of the barn owl. A second system combining innovative sensors and analog computing is presented in the third chapter, dedicated to artificial olfaction and on-line learning. An RRAM array is endowed with the ability to learn from its incoming stimuli thanks to a circuit that implements the Delta learning Rule. Finally, the fourth chapter introduces innovations to conventional neuromorphic computing architectures. RRAMs find an application as plastic elements in neurons and intrinsic neuronal plasticity is combined with spike-dependent-synaptic-plasticity to form an unsupervised learning rule that boosts the performance of spiking neural networks. The structure of neurons is then expanded with dendritic circuits, in which RRAMs are used as both synaptic weights and delay elements. The latter enable new capabilities when processing temporal signals and make spiking-neural-networks more efficient and less memory consuming. All the projects in this thesis open to extensions of what hereby presented. Developments in all the areas - design, device performances, algorithms - will enable to radically improve the results achieved in the different projects. It is with this hope that the work presented in this thesis aim to be the basis for a new wave of neuromorphic systems.

The thesis concludes with a vision for analog, bio-inspired in-memory computing. The field is moving rapidly and major actors in the market are joining to explore new solutions. It is difficult to make long-term predictions but it is tangible that the subject is growing and the future looks bright ahead.

Chapter 2

Building a Spiking Neural Network with RRAM

This chapter presents the process of building a Spiking Neural Network with analog electronics and with RRAM devices. An introductory section motivates the choice of such a computational model and highlights its strengths as well as its weaknesses. The chapter is divided into three parts. The first introduces the Neuromorphic Hardware Calibration procedure which proposes an Off-Line learning scheme for SNNs built with analog circuits. A thorough characterization of fabricated analog circuits and RRAM devices assesses the performance of such architecture. Then, the building blocks of a SNN designed in 130 nm technology are presented, ranging from the small analog circuits that implement neurons and synapses, to the RRAM array and to the assembled system. The last part of the chapter is dedicated to the expansion of the work to a larger scale SNN and the presentation of the Mosaic concept.

2.1 Motivation

Edge AI promises to bring the computational power of Machine Learning and Deep Learning to mobile devices. Embedded computing devices bring several advantages, among which privacy, energy efficiency and adaptability to the environment. Privacy is potentially threatened by cloud computing, as data collected by users has to be handed over to a server in order to be processed and analysed. Communicating data is not energetically convenient either, as one would have to exploit - for example - a wireless infrastructure like 4G or 5G. At last, cloud servers are generally utilized by a large number of users and can't be optimized to the need of the particular use-case. All these problems are solved by moving computation closer to the user and, in particular, to the sensor producing the data. This is why it is called near- or in-sensor computation. The advantages of such a

practice are obvious: the data do not leave the device which produced them, avoiding communicating to third parties for processing and analysis; data movements are thus minimized and the energy efficiency is radically improved.
Selecting what model is best suited for an Edge AI application is a difficult choice depending on what task is to be performed, what sensor is feeding the model and which hardware constraints are imposed by the environment. This is where the field of Edge AI branches into multiple subcategories, depending on the hardware platform running the machine learning models. The main categories are developed around: I) microprocessors and microcontrollers, II) digital application-specific integrated circuits (ASICs) and III) custom analog solutions. The I) class is certainly the most popular and commercially exploited, but the least advanced in terms of power consumption and computational capacity, while ASICs (II) are slowly taking over as they allow to optimize computation for a given application. Systems based on the I) (and sometimes on the II) class too) are conventional Von-Neumann architectures, where memory and computation are physically separated and energy consumption is dominated by memory access. It appears evident that moving beyond from the Von-Neumann architecture represents the most urgent structural change in Edge-AI to reduce energy consumption. In-Memory computation is a novel concept of computing based on the utilization of memory elements directly for the computation. This paradigm is particularly useful for neural network applications where large weight matrices are involved in matrix-vector-multiplications (MVMs). RRAMs, as well as other Non-Volatile-Memories (NVMs), are particularly suited to perform In-Memory computing (IMC), as they are compatible with the Back-End-of-Line (BEOL) of a CMOS process. Many groups have reported the energy benefit of overcoming the Von-Neumann architecture with Non-Volatile-Memories [73, 124, 125, 126, 127, 128]. The solution presented in this chapter locates in the III) class and it's an analog, RRAM-based Spiking Neural Network (SNN).

The presented system involves 3 technologies that are highly promising for Edge AI and that constitute a departure from a classical approach to computation. It is important to point out why a combination of these technologies is crucial to build an extremely efficient inference engine for Edge AI.

- **Why analog electronics?** The proposition is that the information coming from a sensor should not be translated to the digital domain, it should rather be computed in the analog domain. Translating analog information to the digital domain always bears an energy cost, which is not devoted to computation. This energy cost becomes even more problematic when the sensory information has a relevant temporal component and a periodic sampling of the signal is necessary. Analog computation avoids the cost of periodic sampling and translation of the information.

- **Why RRAMs?** RRAMs are analog, non-volatile memories that can store up to 3-bits of information [129] per device. RRAMs can be grouped in arrays that perform

the Matrix-Vector-Multiplication, ubiquitous in machine learning, very efficiently [130, 131, 132, 133, 27, 134, 135]. RRAMs represent a competitive alternative in memory density to more conventional DRAMs and Flash NANDs, and don't require a voltage (and static power consumption) to maintain their state. This makes RRAMs a perfect fit with SNNs, where memory is read asynchronously and sparsely in time and space.

- **Why a Spiking Neural Network?** From the point of view of Edge AI engineering, SNNs promise to improve energy efficiency by compressing the activation function of neurons to a delta function, thus minimizing the communication between neurons and making the model robust to noise and weight variability [136].

The 3 technologies mentioned above seem to be the perfect fit for each other. In fact, the proposed analog RRAM-based SNN solves two major issues of more conventional RRAM-based accelerators, that are analysed hereafter.

2.1.1 The limitations of RRAM-based ANNs

SNNs solve the problem of high column current density in RRAM arrays. This issue limits the scalability of the ANN approach. As visually described by Fig. 2.1 a, ANNs activate all the rows of the array at once, thus producing high output current at each column, and the overall power budget increase linearly with the number of devices being read (i.e. number of activated rows). Another limitation is due to the conventional hybrid analog/digital nature of ANN accelerators, which perform the Matrix-Vector-Multiplication operation in the analog domain, exploiting RRAMs, and then compute the neuron activations and communicate data in the digital domain. The overhead circuits required to convert back and forth between digital and analog domains - Digital-to-Analog (DAC) and Analog-to-Digital (ADC) circuits - significantly increases the area and power consumption. SNNs computation, instead, is sparse in time, so the number of activated rows at any instant of time is very small, significantly reducing the current and power consumption at each column (Fig. 2.1 b). Moreover, analog neurons and synapses in SNNs do not require DACs and ADCs, resulting in a further reduction of energy consumption and area [95].
These points don't configure the proposed solution as ideal for all Edge AI applications, they rather frame analog RRAM-based SNNs as the optimal solution for a particular class of cases. The perfect use-cases are the ones where maximal energy efficiency is required (i.e. the power budget is as low as $<$1mW) and the input signal features relevant temporal components and is sparse in time and space (input dimension). For other conditions and applications, other solutions might be preferred. For example, convolutional neural network (CNN) ASICs are the most popular solution for image processing and in embedded applications [137]. ASICs running particular recurrent neural networks (RNNs) and CNNs are the best choice for key-word spotting and speech recognition [138], and biomedical signal processing and classification [139].

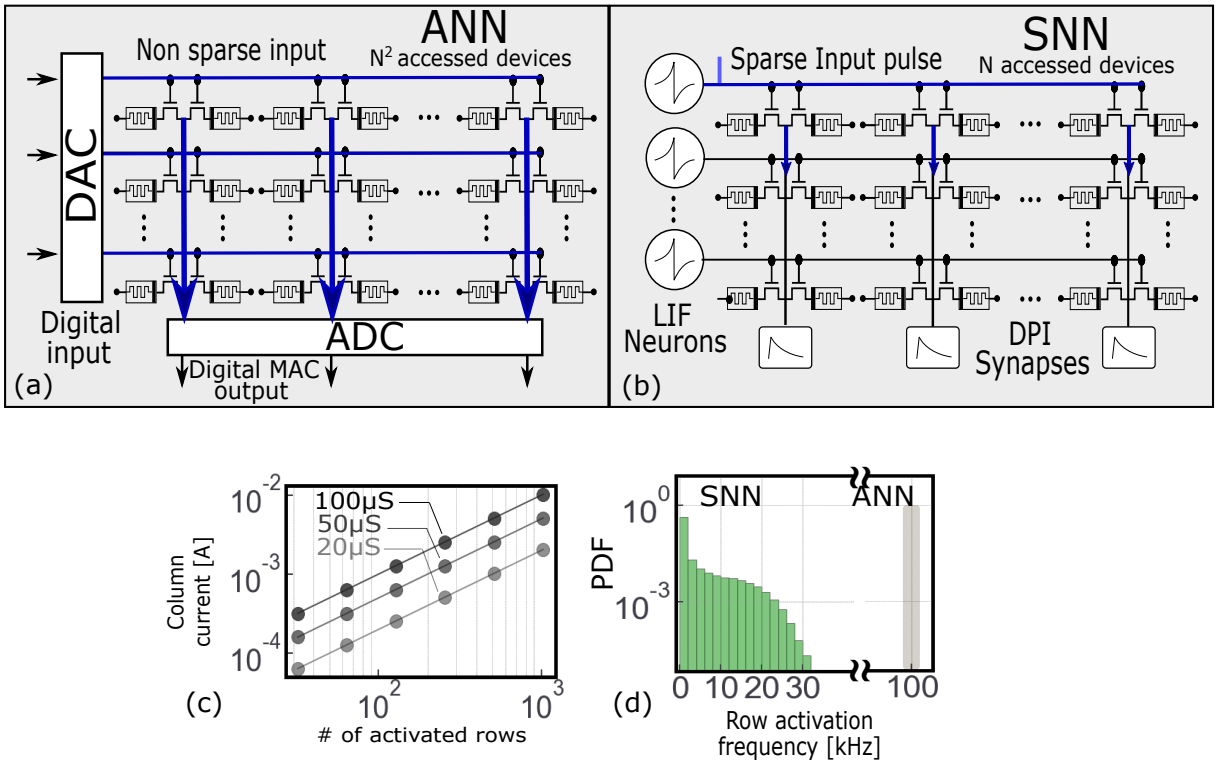


Figure 2.1: RRAM crossbar arrays in artificial neural network (ANN, a) and spiking neural network (SNN, b). ANNs conventionally activate the whole array to make the MVM operation, yielding high current density that limit scalability of the array itself. ANNs also require DACs and ADCs to convert between the digital input/output information and the analog nature of RRAM in-memory computation. SNN solves both issue as it computes with sparse temporal events processed by analog circuits, Leaky-Integrate-and-Fire neurons and Differential-Pair-Integrator synapses. (c) Quantification of current magnitude per column for different average RRAM conductance. (d) Visualization of the temporal sparsity of SNN, as the row access frequency distribution for ANN and SNN. The SNN rarely access the rows of the array with high frequency, lowering the column current.

The limits of our approach are related to the low maturity of the technology. SNN models are rapidly growing in terms of computational capacity and have been demonstrated to reach good results in a large number of benchmarks [112], however still fall short respect to the state-of-the-art deep learning models. Analog electronics holds great promise in reducing the cost of computation, but the realization of an analog chip is much harder than its digital counterpart. Furthermore, both analog electronics and RRAMs are subject to variability which is detrimental to the model's performance: this will be discussed in details later in the chapter. In general, our approach does not fit applications in which the data to be processed is static and dense of information, such as the case of image processing.

In the following, the details of building an RRAM-based Spiking Neural Network processor for Edge-AI are explained. The chapter opens on why choosing Spiking Neural Networks over conventional Artificial Neural Network, it then confronts the problems of RRAM-based analog electronics computation, and then proposes the solution to calibrate the training procedure to the hardware defects. At last, the proposed model is analyzed on its performance and energy efficiency, comparing to other state-of-the-art systems.

2.2 Variability in analog electronics and RRAMs

Analog electronics circuit often utilize transistors in their sub-threshold regime, expanding the range of possible behavior while minimizing drain-source currents, and energy consumption in turn. Analog circuits are appealing for multiple reasons, among which the fact that they directly process natural analog signals, without loss of information in the signal. This is particularly attractive when treating audio and radio information, as information density of analog signals is higher than the digital counterpart. Furthermore, analog circuits can more faithfully *emulate* biological systems and components, such neurons and synapses. This supports the efforts in neuromorphic engineering to design analog circuits that behave similarly to biological circuits. [70, 140].
However, analog circuits also have major flaws which present great challenges in neuromorphic engineering. The main issue is the variability between transistors, amplified by their usage in the sub-threshold regime. Process variation in doping density, oxide thickness and transistor size [141] have a strong impact on the threshold voltage of analog transistors. These defects become worse and worse as one scales down the technology node. Variability of the threshold voltage radically affects the Current/Voltage characteristics of the transistor, issue that spread across analog circuits with multiple transistors. This makes the behavior or neuromorphic circuits slightly unpredictable and rises challenges when designing a large scale system.
Different methods have been proposed to mitigate variability in the behavior of neuromorphic circuits: most concern the design phase [142], other the algorithm that utilize the circuits, like population coding.
It is well known in the literature that RRAMs are stochastic devices [129]. The variability of RRAMs is related to the nanoscopic scale in which atoms are arranged to form or disrupt the conductive filament. The conductance of the filament results being dependent on the programming conditions - which assures a good degree of control over the device - but is inevitably subject to stochastic phenomena. These stochasticity is often described as Cycle-to-Cycle (C2C) and Device-to-Device (D2D). C2C variability is the result of iterative programming of the same device, under the same programming conditions: the stochasticity of the arrangement of the filament under the same electric field results in a distribution of conductance values. D2D variability is observed when a number of different devices is programmed with the same procedure, obtaining a distribution of conductance levels for the same reason mentioned before. Furthermore, RRAMs also present temporal variability, meaning that their conductive state changes over time. The extent of

this temporal variability is studied in the next paragraph. These phenomena have to be accounted for when utilizing RRAMs for Machine Learning or neuromorphic computing.

2.2.1 Heterogeneity in Neurons and Synapses

To assess the variability of analog circuits, we designed, fabricated, and tested analog CMOS-based LIF neuron and DPI synapse circuits (Fig. 2.2) built in 130 nm technology. The LIF neuron (Fig. 2.2a), inspired by [70] receives input pulses (Vin) and weights them with a DPI circuit biased by Vgain. Input current pulses are accumulated in the membrane capacitor Cmem, increasing the membrane voltage Vmem. At the same time, a small amount of current leaks from the transistor biased by Vlk, which determines the time constant of the neuron. At the crossing of a threshold voltage (the Inverter threshold, half of the supply voltage), a voltage pulse is elicited at the output (Vout). This voltage pulse charges the refractory period capacitor (Crp), coupled to a transistor biased by Vrp, determining the refractory period. During the refractory period, a N-transistor is open, discharging Cmem and impeding input currents to charge the membrane capacitor.
The DPI synapse (Fig. 2.2b) is a circuit presented in [140], in this case modified to integrate an RRAM as synaptic weight. Input signal are presented as voltage pulses (Vin) and some current is read from a RRAM device. The magnitude of the current depends on the conductance of the RRAM. The input pulse is weighted by a DPI, biased by Vthr. Later, the current discharges a synaptic capacitor Csyn, coupled with a leakage transistor controlled by Vtau. This latter bias modulates the time constant of the synapse. The voltage of the Csyn capacitor is the output of the circuit.
Both circuits have been measured to verify their correct behavior. For the LIF neuron (Fig. 2.2c), the experiment consisted in monitoring the response of a neuron's membrane voltage (Vmem in dark blue) to a train of input pulses (Vin in black). The membrane capacitor reacts to each input by increasing its voltage, up to the point in which it overcomes the threshold and emit an output spike (Vout in light blue). The experiment to assess the correct behavior of the DPI synapse (Fig. 2.2d) consists on monitoring the synaptic voltage response to a single input voltage pulse at t=0s. The Vsyn potential depolarizes with a magnitude proportional to the RRAM conductance, and then relaxes back to the resting potential.
A similar experiment is repeated for both the neuron and the synapse, analysing the effect of the leakage bias (Fig. 2.2e,f). Modulating the Vlk (neuron) and Vtau (synapse) biases, the leakage rate is controlled, resulting in different time constants. The measurements have been repeated over 100 samples and the time constant extrapolated from the response of Vmem/Vsyn in time. For both neurons and synapses, the variability in the time constants is about 30% in standard deviation over the mean, while the mean of the distribution can be easily controlled in the $[10^{-1} - 10^{2}]$ ms range.

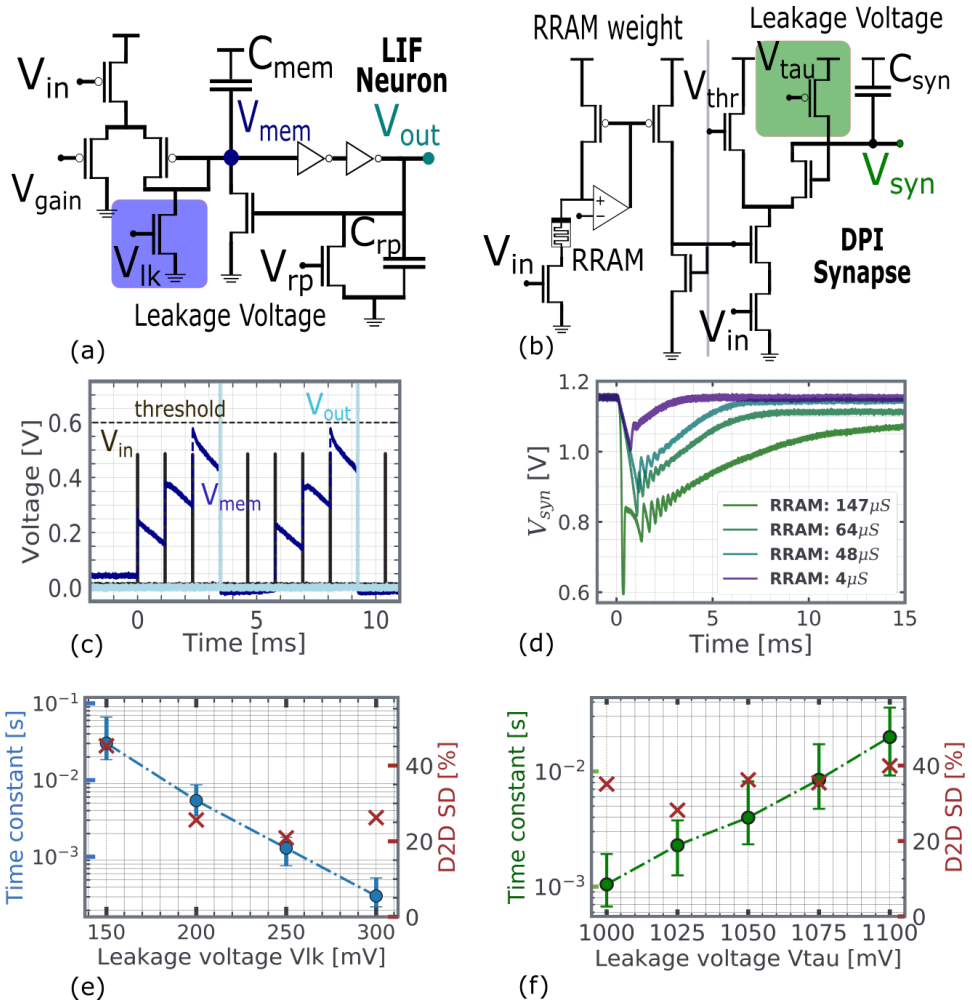


Figure 2.2: Leaky-Integrate and Fire (LIF) neuron circuit (a) and Differential-Pair-Integrator (DPI) RRAM synapse (b) circuits. The circuit have been designed and fabricated in 130 nm technology. (c) Test of the LIF Neuron with an input spike train (black) at 1 kHz: the membrane voltage (dark blue) increases upon the arrival of each input spike; when crossing the threshold voltage, the neuron produces an output spike (light blue). (d) Test of the depolarization of the DPI synapse; an input spike is presented at time 0 s and the voltage on the capacitor is recorded; the amount of depolarization depends on the conductance of the RRAM, reported in the inset. (e,f) Time constant of the LIF and DPI synapse as a function of the V_{lk}/V_{tau} (neuron/synapse) biases. To measure the time constant, the circuit response to a single input spike is recorded. The bias voltage V_{lk}/V_{tau} controls the leakage rate of the capacitor, determining the time constant of the circuit.

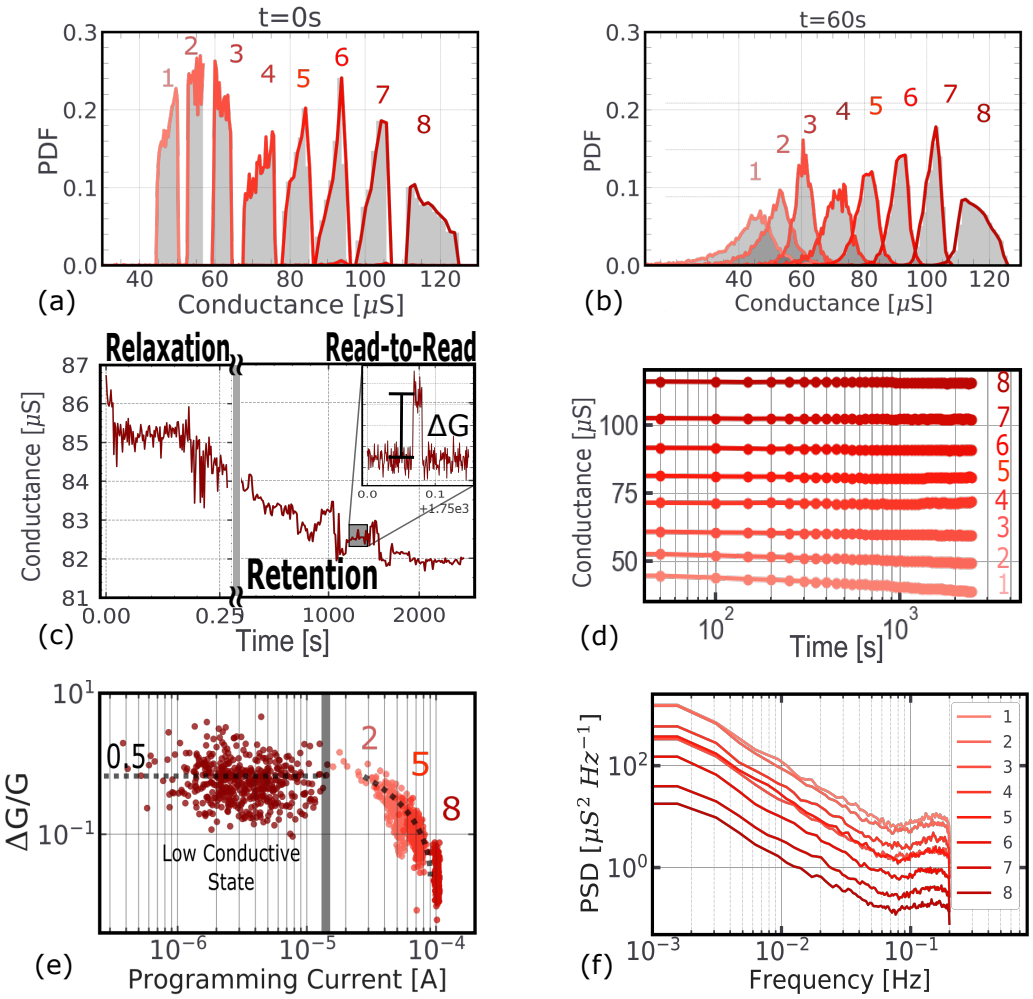


Figure 2.3: Variability in RRAMs. (a, b) Multilevel programming achieves 8 conductance levels. Right after the programming operation, at t=0 s, the levels at tightly distributed and non-overlapping. After some time, at t=60 s, distributions have widened due to Relaxation and are overlapping. (c) Temporal evolution of the single RRAM device as a function of time, distinguishing between different variability forms: Relaxation, Retention and Read-to-Read noise. (d) Retention: the mean of distribution for each of the 8 levels is measured over time; the levels 1 and 2 are less stable and slowly drift along time. (e) Read-to-Read noise: measured $\Delta G/G$ as a function of the programming current (ΔG is due to RTN and is defined in (b)). (e) Power Spectral density of the noise in the 8 conductance levels,showing a 1/f spectrum.

2.2.2 Variability in RRAMs as Synaptic weights

RRAMs hold the potential to be high density non-volatile memory elements. However, only an appropriate programming procedure and an accurate analysis of their stochasticity allow to exploit them in In-Memory neuromorphic computing. To achieve maximal memory

density, a 4kb array of 1T1R devices is programmed with a multilevel smart programming procedure. This procedure described in [129] allows to obtain 8 conductance levels per RRAM device (Fig. 2.3a). The levels are all well separated from each other and narrowly distributed at t=0 s, right after programming. However, RRAM suffer from temporal variability as the conductive filament is re-arranging itself in different configurations over time, leading to a variation of conductance [143]. This is why measuring the same 4kb array 60 s after programming, the 8 conductance levels have widened and collide with each other.
RRAMs show 3 degrees of temporal variability that take place at different time scales (Fig. 2.3). **Relaxation** takes place just after programming (milliseconds) and broadens the conductance distributions (Fig. 2.3c). **Retention** causes long-term (hours) variation of the conductance. The lower conductance levels shift their average toward lower values (Fig. 2.3d). **Read-to-Read** (R2R) noise does not affect the shape of the conductance distribution, although when looking at individual devices there are fluctuations around the average due to reading disturbances and Random Telegraph Noise (RTN) (Fig. 2.3e). We evaluate the RTN component in R2R via the $\Delta G/G$ figure of merit (Fig. 2.3e), measuring the conductance jumps $\Delta G/G$ due to RTN. The result is in line with the literature [143]. Finally, the Power Spectral Density of the 8 conductance levels shows that the amount of noise is inversely proportional to the conductance and is general of the 1/f type (Fig. 2.3), as also observed in [144].

2.3 Neuromorphic Hardware-Calibrated Off-Chip training

Variability is a dangerous defect for neuromorphic analog system, which, if not treated correctly, results in unpredictable behavior and poor performance. Tackling variability is a priority when designing and deploying neuromorphic circuit and this is demonstrated in this section. The latest techniques to train Machine Learning models can be transferred to neuromorphic circuits. The benefit is that gradient-descent methods can work around the defects of the hardware and cope with variability of both analog circuits and RRAMs.

2.3.1 Off-Line learning on SNNs

SNNs are powerful models mimicking the computational principles of the brain, in which communications happens via stereotypical asynchronous voltage pulses called spikes. SNNs models are formed by different spiking neurons, whose activation function is a delta function, connected by synapses, whose role is to route and weigh the spikes across the network. SNNs models can be configured in several architectures, some of them inspired by biology (Liquid State Machine [145] and Attractor networks [146]) some other borrowed from deep learning (feed-forward, CNNs and others). The most common architecture

for SNN is the recurrent network (RSNN) where a hidden recurrent pool of neurons has recurrent connections between each other. This closely resembles the Liquid State Machine, but in the RSNN the recurrent connections are learnt and not randomly established. From the point of view of machine learning, a RSNN (Fig. 2.4a) can be formalized as a Artificial Recurrent Neural Network (RNN) in which artificial neurons are swapped by more biologically plausible neuron models. In this chapter, Leaky-Integrate-and-Fire (LIF) neurons are assumed as the standard choice for all the models.

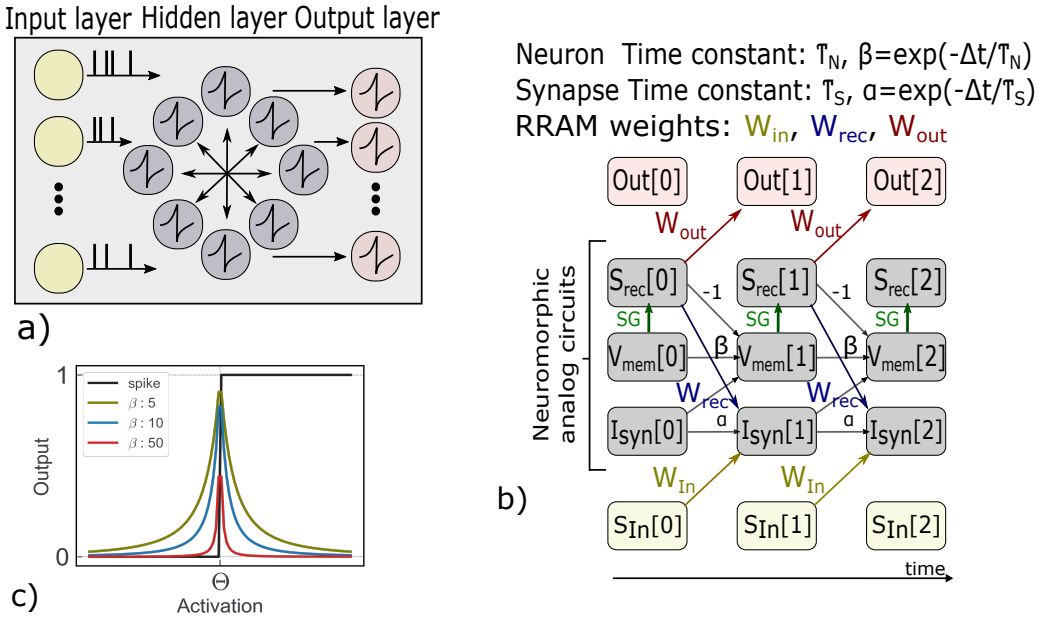


Figure 2.4: Spiking Neural Network offline learning. a) SNN model depiction with the recurrent connections in the hidden layer. b) Computational Graph of the RSNN with input cells in yellow, hidden cells/neurons in grey, output cells in red. W_{in}, W_{rec}, W_{out} are the input, recurrent and output weights respectively. Recurrent connections are depicted by the down-facing arrows from S_{rec} - the neurons' spikes - to the I_{syn} cell, the synaptic dynamic state. SG in green stands for Surrogate Gradient. c) The activation function of hidden cells (neurons) is the Heaviside function with Surrogate Gradient, which is a step function for the forward pass (inference) and a smooth, differentiable function for the backward pass (gradient descent learning). Here, the differentiable function for the backward phase is reported for different β coefficient. Activation is the neuron's membrane voltage for the physical realization of the circuit.

As described in the Introduction, several techniques are used to train SNNs for a given task, ranging from STDP to ANN-SNN conversion, to biologically inspired approximations of gradient descent [115]. We propose the Surrogate Gradient (SG) technique as the optimal choice for Off Line learning: SG assures the best performance for SNN in a wide variety of tasks and benchmarks, and it allows for maximal model flexibility and

adaptability to the specifications of the hardware. This technique simply applies standard gradient descent to SNN, overcoming the issue of non-differentiable activation function of spiking neuron with a trick. In the computational graph of a RSNN (Fig. 2.4b), each step in the graph is differentiable, less for the activation function of neurons. The SG technique enriches the activation function so that it is a normal Heaviside function for the forward pass, but a smooth differentiable function for the backward pass (Fig. 2.4c). Multiple differentiable functions have been tested and give good results, and their role is solely to allow the gradient to keep flowing back in the computational graph. In our case we chose the following smooth function:

$$f(x) = \frac{1}{(1 - \beta|x - \Theta|^2)} \tag{2.1}$$

where x represents the activation or membrane voltage in the LIF neuron, β is the steepness of the function, as seen in (Fig. 2.4c).
The shortcomings of the SG techniques are that SNN training takes a lot of memory in GPU and is in general computational intensive. However, the process of learning is performed only once, before deploying the model to the edge.

The SG technique is particularly interesting for neuromorphic systems as it allows to take into account the non-idealities of the hardware substrate in the learning phase. Based on this principle, the Neuromorphic Hardware Calibration procedure is presented in Fig. 2.5. The procedure starts with the fabricated RRAM-based analog RSNN chip, whose analog circuits are characterized one by one. Heterogeneity of neuromorphic circuits is introduced by assigning each neuron and synapse a different time constant value based on the experimental data, as in Fig. 2.2e,f. Once the hardware heterogeneity is loaded in a server, a computer performs the SG-based gradient descent optimization of the network, depending on the task of choice. The resulting model features floating-point weights with 32bit precision (Fig. 2.5I)). The weights are quantized to 15 levels (4bits), as allowed by 2 RRAMs with different polarization and with 8 conductive levels each (Fig. 2.5II)). At last, the quantized weights are mapped to the RRAM array by selecting the appropriate reading voltage, which acts as a normalization coefficient for the synaptic weights (Fig. 2.5III)). The procedure concludes with the programming operation of the RRAM array in the RSNN chip, which is then ready for deployment.
To evaluate the approach, this procedure is performed for 3 different benchmark tasks with different degrees of temporal structure: MNIST [147] (static visual image of handwritten digits), ECG [148] (heart arrhythmia classification), and SHD [149] (spoken digits). As SNN take input information in the form of spike-trains, a little description for the pre-processing required to feed the dataset to the SNN is provided.

N-MNIST The popular MNIST dataset consists of images of hand-drawn numbers from 0 to 9. In order to feed this dataset with SNN, the static images have to be converted to

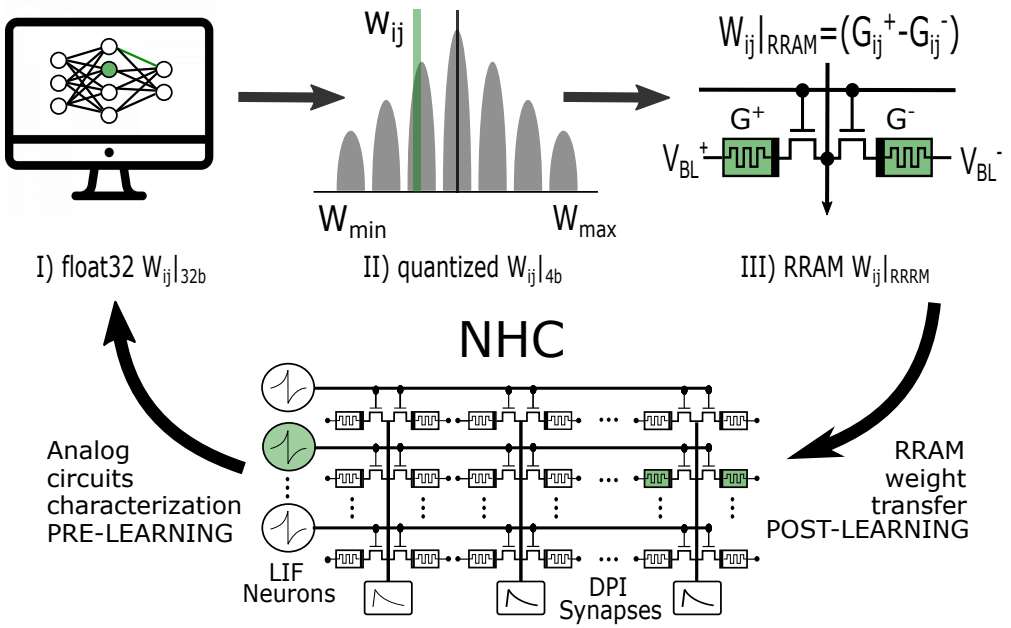


Figure 2.5: Neuromorphic Hardware Calibration procedure. Based on a RRAM-based analog SNN chip, analog circuits are characterized and the resulting data calibrate an Off Line learning procedure in a server (left). I) The weights in the SNN model are trained in fp32 precision. II) Later, the parameters are quantized on N levels, as many as admitted by the RRAMs. III) At last, the quantized weights are converted to RRAMs, where the conversion factor V_{BL} represents the reading voltage of the RRAM. The process ends with the Post-Learning transfer of learned weights to the chip. The NHC can be repeated when the functionalities of the model need to be updated.

dynamical trains of spikes. The conversion method chosen in this case is the time-to-first-spike encoding, where each pixel's intensity is converted with inverse proportion to the time-step of the single spike produced by that pixel channel.

ECG dataset The ECG dataset was downloaded from the MIT-BIH arrhythmia repository [150]. The database is composed of continuous 30-minute recordings measured from multiple subjects. The QRS complex of each heartbeat has been annotated as either healthy or exhibiting one of many possible heart arrhythmia by a team of cardiologists. One patient exhibiting approximately half healthy and half arrhythmic heartbeats was selected. Each heartbeat was isolated from the others in a 700 ms time-series centered on the labelled QRS complex. Each of the two 700 ms channel signals were then converted to spikes using a delta modulation scheme [151]. This consists in recording the initial value of the time-series and, going forward in time, recording the time-stamp when this signal changes by a pre-determined positive or negative amount. The value of the signal at this time-stamp is then recorded and used in the next comparison forward in time. This

process is then repeated. For each of the two channels this results in four respective event streams - denoting upwards and downwards changes in the signals. During the simulation of the neural network, these four event streams corresponded to the four input neurons to the spiking recurrent neural network implemented by the SNN.

SHD dataset Spiking Heidelberg Dataset (SHD) [149], which is a recently-proposed benchmark for SNN, based on the recording of spoken digits from zero to nine in English and German (20 classes). The Mosaic-based model was trained using the same surrogate gradient approach as in the ECG task, and we compare its performance to a software-based RSNN. The Spoken Heidelberg Dataset was created in 2020 with the aim of being a challenging task for SNNs in the class of keyword spotting. Data captured from a microphone is pre-processed with the Lyon-ear model, yielding a N-channels input in the form of spike trains. The dataset is based on the recording of spoken digits from zero to nine in English and German (20 classes).

To solve such tasks, a RSNN as sketched in Fig. 2.4a is used, with two different configurations. **In** is the input size, **Hid** the hidden layer size, **Out** the output size. For the N_MNIST dataset the recurrent connections in the hidden layer are switched off: the network is In 784-Hid 128-Out 10. For the ECG and SHd tasks, recurrent connections are switched on and the network has the form In 4-Hid 128-Out 5 for ECG and In 784-Hid 128-Out 20 for the SHD.
Table I compares the NHC SNNs results with ideal software-based ANNs and ideal SNNs (i.e. with homogeneous time constants). Non-Calibrated (NC) SNN, instead, is the case where the neuromorphic circuit present variability but this is not considered during the learning phase. The effect of variability on the circuits will then be felt during deployment of the neuromorphic chip. As expected non-calibrated SNN chips perform poorly during deployment, as the variability of neuromorphic circuits make the SNN behave differently than expected in the learning phase. This reinforces the need for the NHC procedure.
Homogeneous (Hom.) SNN represents the unrealistic scenario in which neuromorphic circuits do not suffer from variability. Learning and deployment share the same ideal circuits and performance of the SNN are excellent, even when comparing to ANNs. converting such models to 4bits quantized weights leads to a small drop in accuracy, due to the lower precision of the weights.
NHC SNNs score the highest accuracy across spiking models. High precision weights (fp32) allow to maximise performance. Little accuracy is lost when quantizing the weights to 4 bits. Further accuracy drop is caused by the transfer to an RRAM array, which introduces variability in the weights over time. Indeed, accuracy drops as a function of time.

Impact of neuromorphic circuits variability on performance NHC good performance not only highlight the need for a calibration procedure to account for neuromorphic

circuits' variability, it also reveals an unexpected point: variability is beneficial for the network performance. In similar experiments, [152] show that heterogeneity in neurons leads to better performance. This phenomenon could be explained by the richer temporal dynamics offered by heterogeneous networks of neuron, exploited for tasks with relevant temporal features like ECG and SHD. Such a result represents an incentive to neuromorphic engineers to embrace variability in their circuit, trying to exploit it as a feature for their systems.

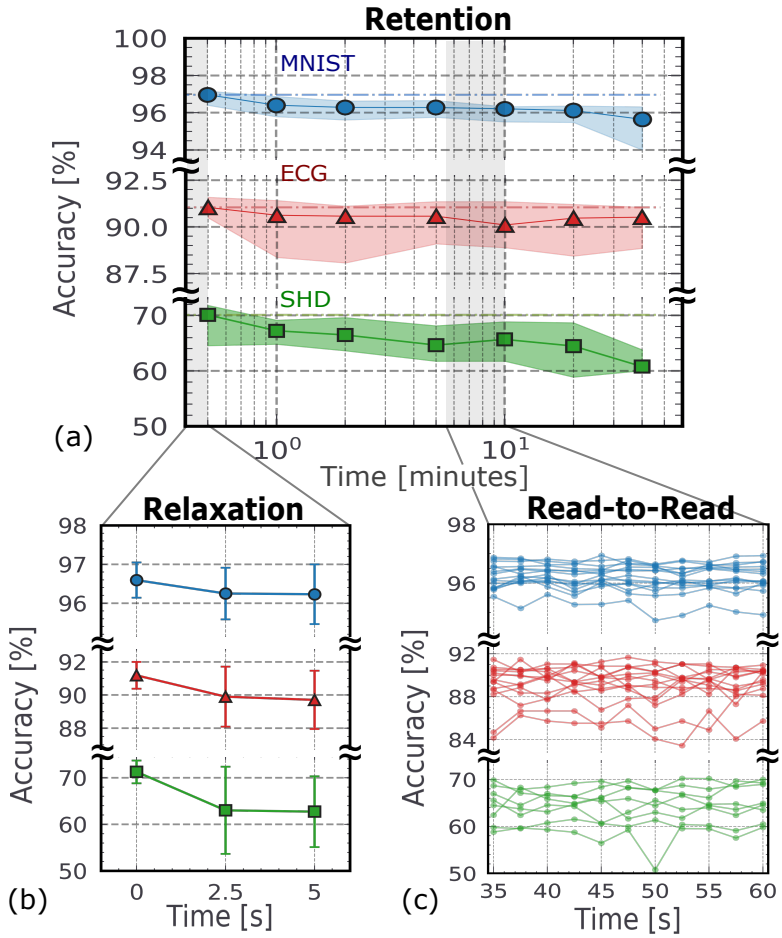


Figure 2.6: Accuracy for the three benchmark tasks, tested with the RRAM array measured across time. (a) Data Retention acts over the course of hours, reducing accuracy. Network used for easier tasks like MNIST and ECG retain accuracy better than those used for harder tasks like SHD. (b) Relaxation induces an accuracy drop immediately after programming. The following measurements show much smaller accuracy drops. (c) R2R causes small variations of conductance each time RRAMs are read, slightly perturbing performance.

Impact of RRAM Non-Idealities on performance The RRAMs support up to 8 distinct conductance levels right after programming, enough to avoid relevant performance

	Weights	N-MNIST	ECG	SHD
ANN	float32	97.5%	95.5%	89.0%
NC SNN	float32	90.2% ± 2.5%	63.7% ± 11.2%	58.4% ± 2.9%
Hom. SNN	float32	97.4% ± 0.1%	94.5% ± 1.2%	72.5% ± 2.5%
	4bits	96.7% ± 0.3%	91.4% ± 1.2%	71.6% ± 1.8%
NHC SNN	float32	**97.5%** ± 0.1%	**94.9%** ± 0.4%	**74.9%** ± 0.9%
	4bits	96.9% ± 0.3%	91.4% ± 1.5%	73.2% ± 2.2%
	RRAM t=0s	96.8% ± 0.4%	91.2% ± 0.8%	71.2% ± 2.4%
	RRAM t=5s	96.2% ± 0.6%	90.2% ± 1.8%	67.5% ± 9.3%
	RRAM t=1h	95.3% ± 0.7%	89.9% ± 1.8%	60.4% ± 7.6%

Table 2.1: Accuracy results of different model, NHC SNN included, over 3 tasks (N_MNIST, ECG, SHD). The mentioned models include ANN, Non-Calibrated SNN (NC SNN), Homogeneous SNN (Hom. SNN) and NHC SNN. Results are reported with float32, 4bits and RRAM weight precision. The NHC results for RRAM weight are reported for the cases of weight just being programmed (t=0 s) and after some time from that moment (5 s, 1 h).

drops for the 3 applications of choice, as shown in Table 2.1. However, the main source of accuracy loss stems from the temporal variability of RRAMs. The impact of such variability is studied in Fig. 2.6. Poor Retention in RRAMs impacts the lower conductance levels of RRAMs causing a slow drift of weights over time. This is mirrored in the accuracy of the SNN models, which slowly decreases. The loss is amplified for difficult tasks such as SHD, while less impactful for MNIST and ECG.
Relaxation causes an immediate decrease in performance (Fig. 2.6b). This is because the change of conductance is higher right after programming, when the state of the RRAM's filament is more unstable. After that first drop, RRAMs land on more stable states and accuracy of the SNN changes less over time.
R2R noise slightly varies the conductance values at each inference operation (Fig. 2.6b). Due to the peculiar characteristics of RRAMs, the conductance state of the devices changes slightly any time they are read and so do the weights of the network. The behavior of the SNN marginally changes every time. This is not problematic, though, as the amount of the RRAM R2R variability does not impact the accuracy too much.

Failures in RRAM-based neuromorphic chips RRAMs and neuromorphic chips can be subject to failures, which threaten the performance of the SNN chip. It is important to know how to deal with such defects so to ensure the longevity of the chip. Failures can occur to multiple components of the chip, for simplicity we limit the analysis to neurons and synapses. 3 different scenarios are proposed. First, the case of RRAM failures: a failure is represented by a device stuck at either low (1μS +- 0.5μS) or high (200μS +-25μS) conductance. The accuracy as a function of the RRAM's failure Rate is shown in Figure 2.7a: SNN models are resilient up to RRAM error rates of 10^{-3}, independently of

the task. Similarly, the accuracy as a function of neuron failures is analyzed (Fig. 2.7b). In this case, a failure causes neurons in the hidden layer to either never spike or to spike at very high frequency (750Hz). Again, SNNs can stand failures up to 10% of neurons (over a total of 128 in the hidden layer), before reducing performance considerably. This happens similarly for the 3 tasks considered.
However, even in the cases of strong damages to the chip, the NHC procedure can be repeated to recover good performance. The off-line learning phase has to be informed with the damaged circuits and RRAMs and SG gradient descent works around the defects and failures and optimizes the chip for the task of choice. In an experiment, SNN models are artificially damaged with a RRAM error rate of 10^{-2} and the NHC procedure retrains them off line (Figure 2.7c) . MNIST is re-learned with just one learning epoch, while ECG and SHD require a few more epochs to recover. Overall, the performance is almost fully restored in all cases. This ensures a heavily damaged chip is not to be discarded, it is just to be taken more care of, with the NHC procedure.

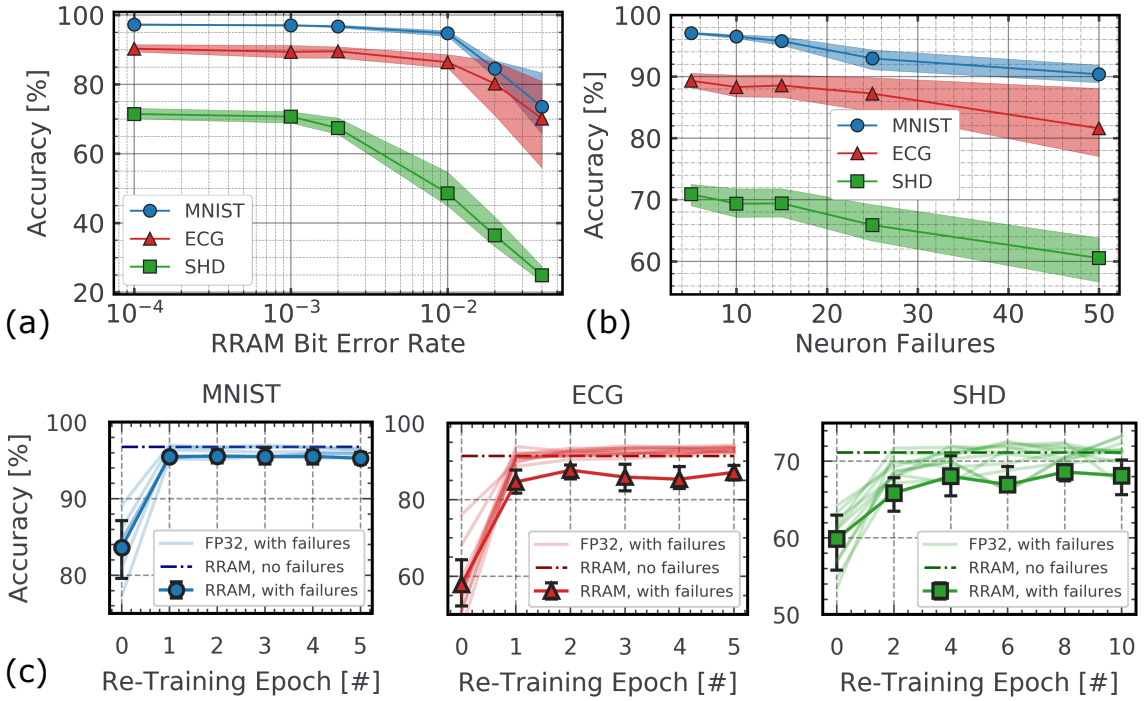


Figure 2.7: Analysis of performance with RRAM failures and re-training taking the failures into account. (a) Accuracy as a function of the Bit-Error-Rate (BER) of RRAM weights. Failures in RRAMs are considered to be the case in which the device is stuck at either low conductance (around 5μS) or high conductance (around 150μs). (b) Accuracy as a function of the failures of neurons, considered to be the case where neurons lose the ability to spike. (c) Networks with high degree of RRAM failures (error rate of 10^{-2}) re-trained considering the weight defects. The NHC helps working around the hardware failures to recover good accuracy.

2.3.2 Energy Assessment

To assess the efficiency of an RRAM based neuromorphic processor we compare their energy per inference sample with a mixed-signal neuromorphic processor, DYNAP [29]. DYNAP uses similar LIF neuron and DPI synapse circuits, implemented in 180 nm technology, but employs an asynchronous digital communication protocol to implement network connectivity. Address Event Representation (AER) is the name of the digital communication protocol. The energy consumption for the RRAM based system is estimated by means of SPICE simulations and is more than 1 order of magnitude lower than that of DYNAP (Fig. 2.8a). Energy figures are reported per inference operation, meaning that a single data point (MNIST image, ECG heartbeat or SHD digit) is passed through the network and the output is read. Inference takes different time for the different task, hence the choice of energy per inference rather than power consumption.
In the simulations of the RRAM-based SNN, spikes are squared pulses at 1.2 V of peak voltage and 100 ns of duration. Spike-width is a fundamental parameter determining the energy consumption of the system: long pulses drain currents from RRAM and the neuromorphic circuits' capacitor for longer, decreasing the energy efficiency; however, too short pulses are hardly handled by analog circuit affected by variability. Variability in analog circuit affects the correct transmission of spikes, so that a safe - large enough - pulse-width must be chosen.
Energy is dominated by RRAM readings, that access the synaptic weights. Conductance in RRAM is in the range [1-150]μS and such high values produce large currents even when reading the device with as low as 100 mV. However, the energy related to read RRAMs is still about 1 order of magnitude less than that of the communication protocol used in DYNAP (Fig. 2.8b). Analog neuromorphic circuits confirm their ultra-low power nature by consuming less than 600 nW per inference. This is also explained by the great sparsity of SNN computation. (Figure. 2.8c reports the distribution of the frequency of spikes, which in turn access the rows of the RRAM array, for each of the 3 considered tasks. Most of the time, spikes are sparse and peak spike density is rather sporadic and limited to 50 kHz. It is reminded that reducing the number of simultaneously activated rows of the RRAM array, has the added benefit of not overcharging the columns of the array.

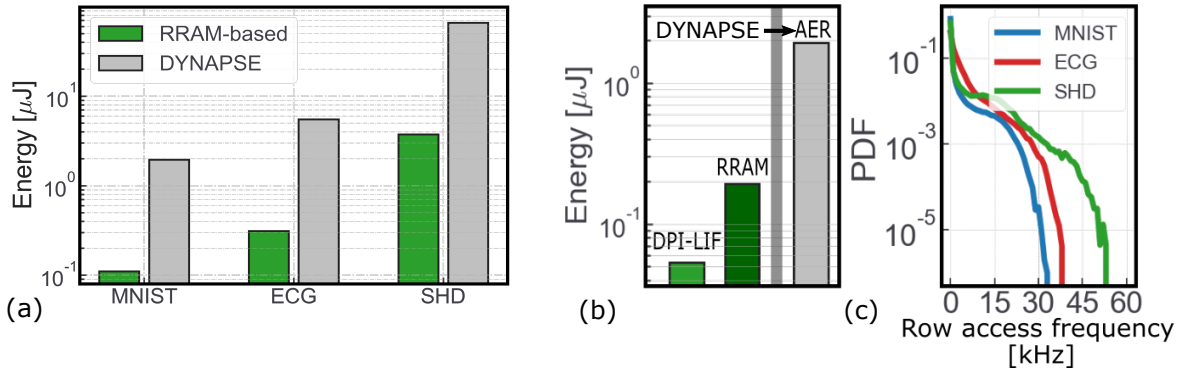


Figure 2.8: Energy assessment of the analog RRAM-based RSNN trained with the NHC procedure. (a) Energy per inference of RRAM-based SNN, compared to DYNAP [29], for the 3 datasets used in this work. The same number of spike is assumed for the two hardware platforms. (b) Energy contributions of the RRAMs and analog circuits (DPI-LIF) per MNIST inference, compared to the routing in DYNAP. (c) Assessment of the sparsity of the network. Assuming to discretize time in bins of 1ms, spikes are gathered in these bins. Each spike results in an access of a row of the RRAM array.

2.4 Design of a RRAM-based SNN

This section of the chapter presents the guidelines to assemble a Recurrent SNN built around a RRAM array and with analog electronics. The system is divided in blocks: RRAM array, Array Periphery and Neuron Periphery. Additional blocks are required to input and output information to/from the system and are only rapidly mentioned. The section analyzes the main blocks composing a RSNN and explains the choices made during the layout phase. A 130 nm CMOS technology node is used, featuring 2 types of transistors: GO1 thin oxide core device, and GO2 thick oxide devices as the input/output option. GO1s are generally used for logics and the analog computation (neurons and synapses) and have a power supply voltage up to 1.2 V; GO2s are used to handle the programming operations of RRAMs, which require higher voltages, up to 5 V. The end of the section shows how the blocks are assembled to form the RSNN chip, which is described in details. The chip includes different versions of the RSNN with different sizes ranging from 8 up to 32 neurons. Future works might extend the network size to verify the limits of the approach when scaling up the system. This design is one of the first fully analog RRAM-based SNN and is not optimized to be the benchmark for energy efficiency and footprint area. It is rather to be analysed as a first prototype, where the concept is implemented, with a big room for improvement on different aspects, among which the main one is certainly the footprint area. Also, scaling to a more advanced technological node will allow to improve in this sense, and to reduce the energy consumption by a relevant amount.

2.4.1 RRAM Array

1T1R devices are the core elements of an RRAM array. RRAMs are tiny nanodevices which are appealing as dense memory elements. However, the limiting factor for footprint area in 1T1Rs is the access transistor, which is utilized as a switch to address the device during the programming, forming and reading phase. Such transistor then has to withstand the current flowing through the device and the high voltages during forming: it happens to be relatively big compared to the RRAM. To comply with the current and voltage requirements, the transistor is of the thick-oxide type (GO2) with 1μm width and 500nm length. The RRAM has lower sizes, at 300x450 nm. In this design, area optimization was not the focus, as the chip was produced as a demonstrator of the RRAM-based SNN concept. It follows that the 1T1R device designed for the project is not optimized for density, with a pitch of 6.56μm both in the x and y dimensions. The large dimension of the 1T1R cell is chosen to ease the layout of more complex blocks at the periphery of the array, choosing a common pitch for all the blocks in the design.
The RRAM array block is the one responsible for the Matrix-Vector-Multiplication operation (MVM) which occurs with temporal sparsity in SNNs. To benefit from the parallelization of the operation, RRAMs have to be arranged in a crossbar architecture, where source lines and bit lines are shared across multiple devices, but are perpendicular to each other. Word lines can be shared in either of the directions. Conventionally, the source line is assumed as the top-electrode of RRAMs and is the common terminal - from a line of N RRAMs - on which currents are summed. On a ANN, inputs are fed to the array through the bit lines, with all the word lines open and result with the MVM to occur by summing currents along the source lines. On SNN, inputs are stereotypical voltage pulses with fixed voltage, normally set at the supply level. A different wiring of the 1T1R devices is preferred to perform the MVM sparsely in time. Bit lines are all set to either ground or a fixed voltage V_{BL}, while source lines are all fixed at the same potential V_{SL} defining the reading voltage $V_{read} = V_{SL} - V_{BL}$. Source lines and word lines run perpendicularly across the array. Spikes activate the word lines with a voltage pulse and allow a current to flow through a single device at the time, unless two or more spikes are synchronous. Each device contributes for a current unit summed on the source line, performing the temporal sparse MVM.
The density of the RRAM array is an aspect which deserves special attention. While it's trivial to prefer a more compact design, selecting the pitch of the array needs to take into account more factors than the size of the RRAM unit cell. The array has to be connected to both the Array Periphery - which is used during the RRAM programming phase - and the Neuron Periphery blocks, working during inference. Matching the pitch of the array to those of the other blocks is fundamental not to incur to area inefficient routing. For this reason, a pitch of $6.56\mu m$ is chosen: this allows the area-hungry capacitors in the DPI synapse and LIF neurons to fit in a row of the array. However, it is recalled that RRAM arrays might reach much higher density when used as binary memories in conventional Von-Neumann architectures [76].
The resulting RRAM unit cell is depicted in Fig. 2.9, along with the full RRAM array. The

final dimension of the array are [209x223] μm. One notes that the array is not squared, it has 32 rows and 34 columns. This is because 32 rows are dedicated to connect to a neuron each, part of the hidden recurrent layer. The input is fed to the network via additional columns of the array, implementing the input layer of the network. In this case, 2 columns are dedicated to external inputs, 32 to recurrent connections from the neurons, making a total of 34 columns.

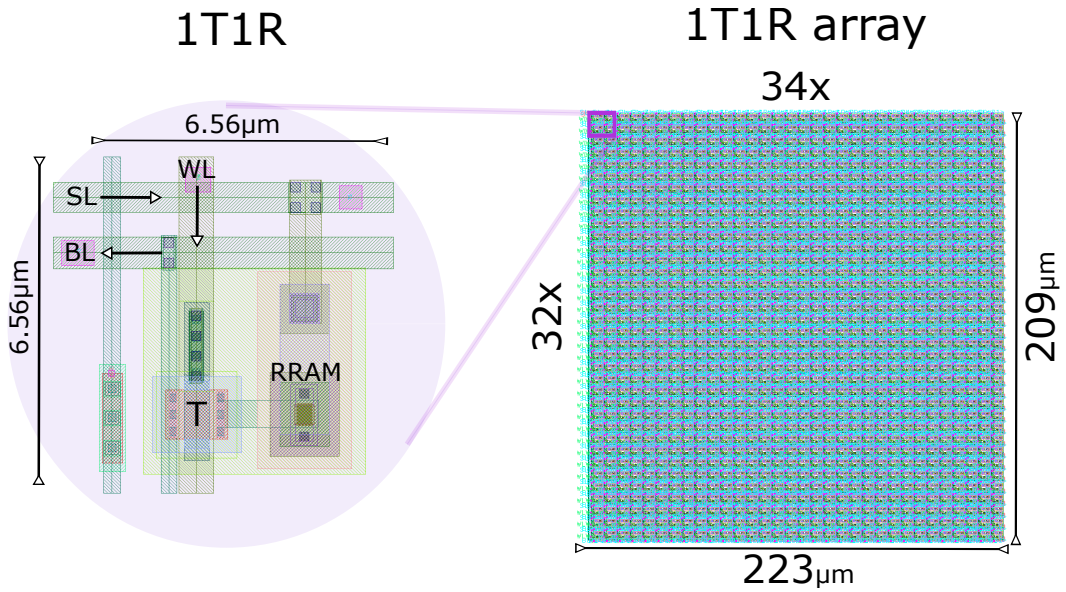


Figure 2.9: RRAM array layout. (Left) The bit-cell for the 1T1R device is designed with a pitch of 6.56μm both horizontally and vertically. Word lines (WL) run vertically, while source and bit lines (SL, BL) are horizontally disposed. The SL connects to the top electrode of the RRAM, BL to the bottom electrode. The access transistor is indicated by a "T" and is 1μm wide and 500 nm long. (Right) The array is made of 34 columns and 32 rows of 1T1R devices and measures [209.2x223]μm^2.

2.4.2 Array Periphery

RRAMs need to be accessed from dedicated pads to program them into the desired state. To achieve this a peripheral circuit has to be designed to access one or more device at the time. The solution of choice is a conventional Scan Chain made of flip-flop circuits whose output is buffered and enabled by a dedicated signal (Enable, EN). the enabled signal is then the selector of an analog multiplexer (MUX) which connects to either the BL or the SL of the RRAM array. The other end of the MUX drives the wire to two pads, one is operated at the programming voltage (Prog) and other is left floating (Float).
The scan chain allows for parallel programming and reading, while optimizes the number of pads required to address one or more RRAM cells. It works as follows: a Clock keeps track of time and is presented to all the flip-flops in the chain; an input binary sequence is presented to the first flip-flop at the D port. At each clock cycle, the D state of the

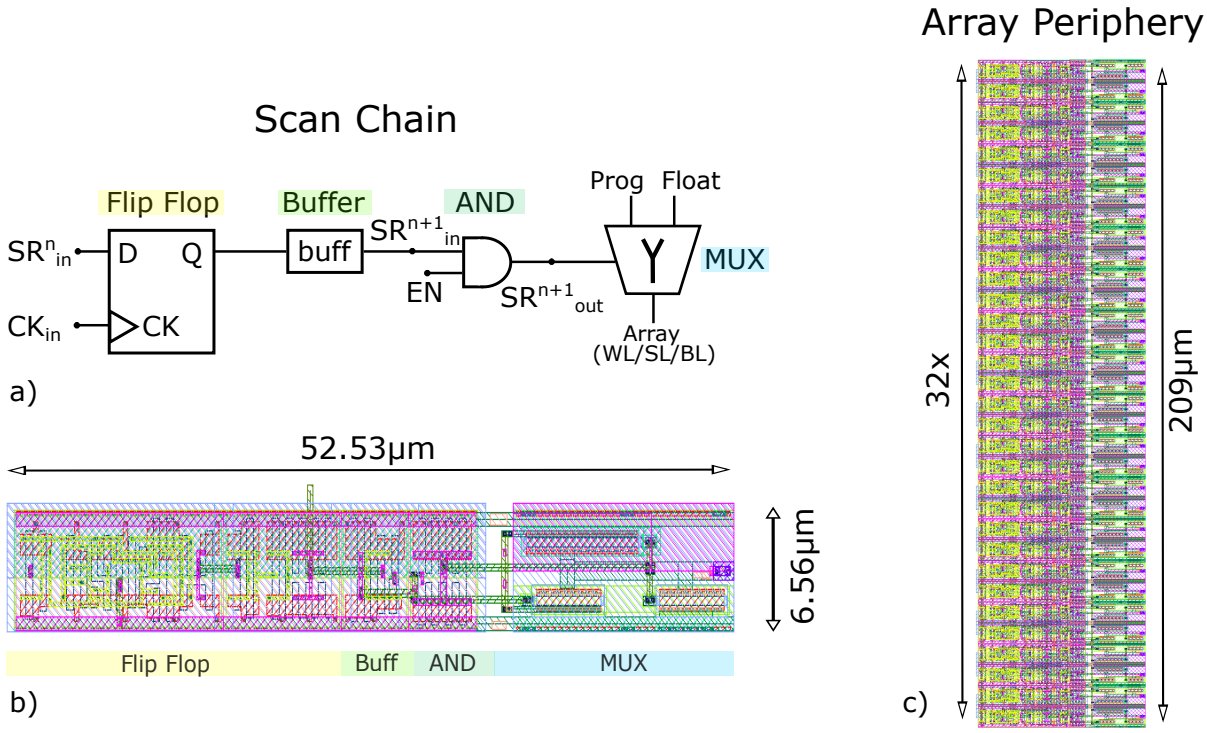


Figure 2.10: Periphery of the 1T1R array. a) Scan Chain block composed of a Flip Flop, a Buffer, a AND and a MUX. b) Layout of the Scan Chain block with all the components respecting the 6.56 μm pitch imposed by the RRAM array. c) 32x Periphery block dedicated to controlling either Word, Source or Bit Lines of the array.

flip-flop is output as Q state and propagated to the next element of the chain. If one has N columns/rows to select, one needs N clock cycles and a sequence of N bits (with appropriate synchronization with the clock), to fill the scan chain. The elements of the sequence filled with a logical 1 (high voltage) are enabled by the EN signal through an AND port. In this way, the logical 1s in the sequence select which columns/rows to program or read.
The circuit schematics is presented in Fig. 2.10. The sensitive element in this circuit is the MUX, as its terminals have to sustain the programming and forming voltage of RRAM, which can approach 5V. For this requirement, GO2 transistors from the 130nm design kit are chosen: these transistors possess a thicker oxide and need to be adequately sized in order to allow the large programming currents to flow. This is also why a large pitch of $6.56\mu m$ is chosen. For simplicity, also the rest of the components in the scan chain are implemented with GO2 standard cells. The resulting element of the scan chain is $52.53\mu m$ wide. The scan chain element are stacked in blocks of 32 for source line and bit lines, in blocks of 34 for word lines.

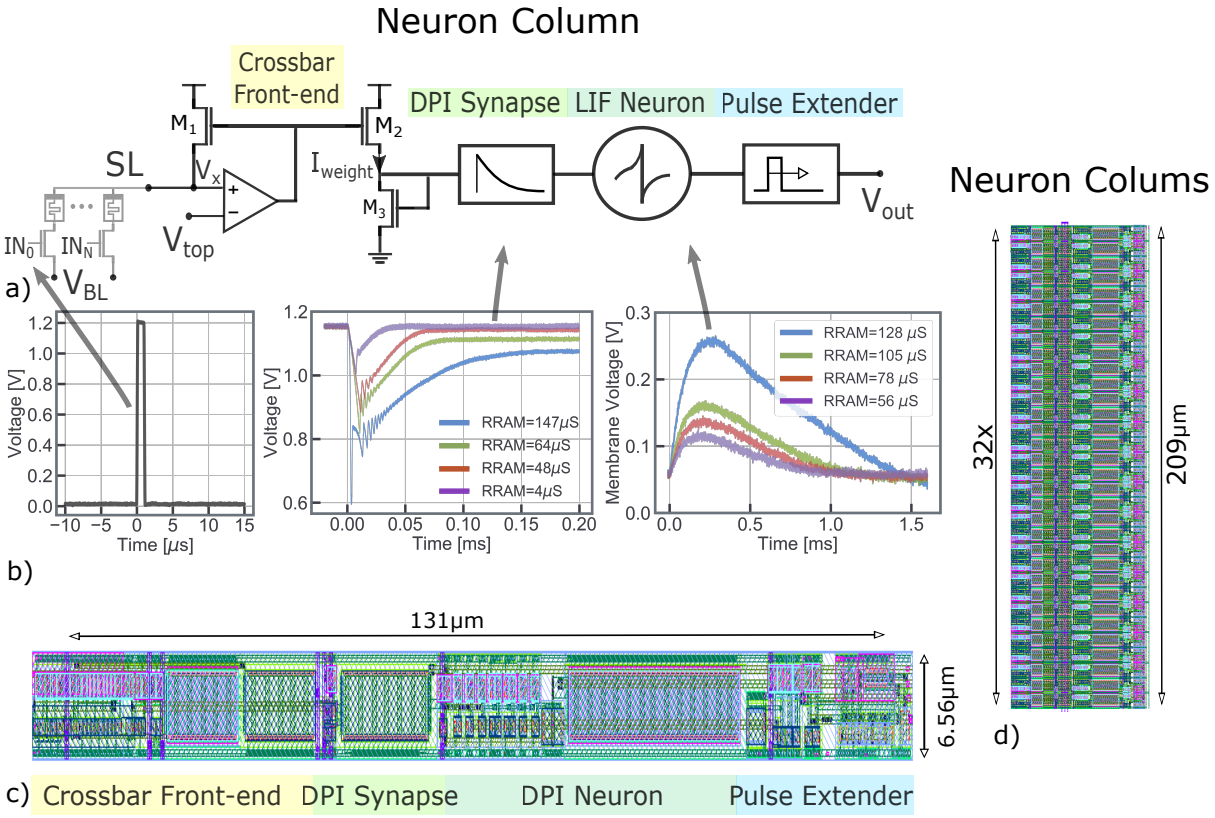


Figure 2.11: Neuron Column design. a) The Neuron Column is composed by a Front-End (Sense Amplifier) connected to the 1T1R array, a DPI Synapse, a LIF Neuron and a Pulse Extender. b) Experimental characterization of the neuron column circuit. An input is presented to the gate of the n^{th} 1T1R device, generating a current I_{weight} from the Front-End circuit. This current stimulates the DPI synapse's depolarization (middle plot), as a function of the RRAM conductance. In turn, the neuron is excited by an amount proportional to the RRAM conductance (right plot). c) Layout of the Neuron Column, where all the components respect the pitch of $6.56\mu m$. The large capacitors in the DPI synapse and LIF neuron result in a narrow rectangular shape. d) The array of neuron column is obtained by stacking each circuit on top of each other, 32 times.

Neuron column circuit The neuron column features a front-end circuit in Fig. 2.11 which reads the conductances of the RRAM devices. The RRAM bottom electrode has a constant DC voltage V_{bot} applied to it and the common top electrode is pinned to the voltage V_x by a rail-to-rail operational amplifier (OPAMP) circuit. The OPAMP output is connected in negative feedback to its non-inverting input (due to the 90 degrees phase-shift between the gate and drain of transistor M_1 in Fig. 2.11) and has the constant DC bias voltage V_{top} applied to its inverting input. As a result, the output of the OPAMP will modulate the gate voltage of transistor M_1 such that the current it sources onto the node V_x will maintain its voltage as close as possible to the DC bias V_{top}. Whenever an input

pulse In_n arrives, a current i_{in} equal to $(V_x - V_{bot})G_n$ will flow out of the bottom electrode. The negative feedback of the OPAMP will then act to ensure that $V_x = V_{top}$, by sourcing an equal current from transistor M_1. By connecting the OPAMP output to the gate of transistor M_2, a current equal to i_{in}, will therefore also be buffered, as I_{weight}, into the branch composed of transistors M_2 and M_3 in series. This current is injected into a CMOS differential-pair integrator synapse circuit model [153] which generates an exponentially decaying waveform from the onset of the pulse with an amplitude proportional to the injected current. Finally, this exponential current is injected onto the membrane capacitor of a CMOS leaky-integrate and fire neuron circuit model [154] where it integrates as a voltage. Upon exceeding a voltage threshold (the switching voltage of an inverter) a pulse is emitted at the output of the circuit. This pulse in turn feeds back and shunts the capacitor to ground such that it is discharged. Additionally, a Pulse Extender circuit assures that the pulses emitted by all the neurons in the system have equal pulse-width. This simple circuit has been designed and fabricated in STM 130nm bulk CMOS technology and tested to verify its behavior. A gray voltage pulse is the input of the circuit and applied to the gate of the i^{th} RRAM. The depolarization of the DPI synapse depends on the conductance G_i on which the pulse is applied. In this case, the RRAM conductance is varied in the [4-150]μS range. The time constant is set by the V_{lk} bias and results in some 200μs. The ripples in the synapse's voltage are due to measurements artifacts. The depolarization of the synapse stimulates the membrane voltage. Again, the stronger the conductance of the RRAM, the higher the response of the LIF neuron, as demonstrated in Fig. 2.11, where the conductance is changed in the [50-130]μS range.
The difficulty of laying out this circuit is that of placing the large neuron's and synapse's capacitors, while respecting the pitch of the RRAM array. Again, this is why a too aggressive size of the array results unnecessary or even detrimental for the area footprint of the chip. The pitch of 6.56μm is a reasonable compromise. Nonetheless, the neuron column circuits ends up with a rather narrow rectangular shape. Stacking multiple neuron column circuits results in the Neuron block, which is one of the three building blocks of the RSNN.

2.4.3 Assembling the RSNN chip

When all the components of the RSNN are matched to the same pitch, assembling the chip is relatively easy. Word lines, source lines and bit lines of the array all require their dedicated peripheral array. Word lines are disposed vertically, while source/bit lines are horizontal. It follows that word lines periphery is placed above the array, bit and source lines are on the left and right of the array respectively. Neuron columns connect to the source lines and are thus placed to the right of the SL periphery block. The top level view of the chip in Fig. 2.12 highlights the large dimension of the periphery circuits respect to the RRAM array itself. This is one of the weak points of this approach, where a large periphery circuit is required to apply high voltages (up to 5V) to the devices and to sustain

large currents (up to 300μA per device, with N devices in parallel). The price to pay for the high capability offered by such circuit is that they take up a considerable amount of area.

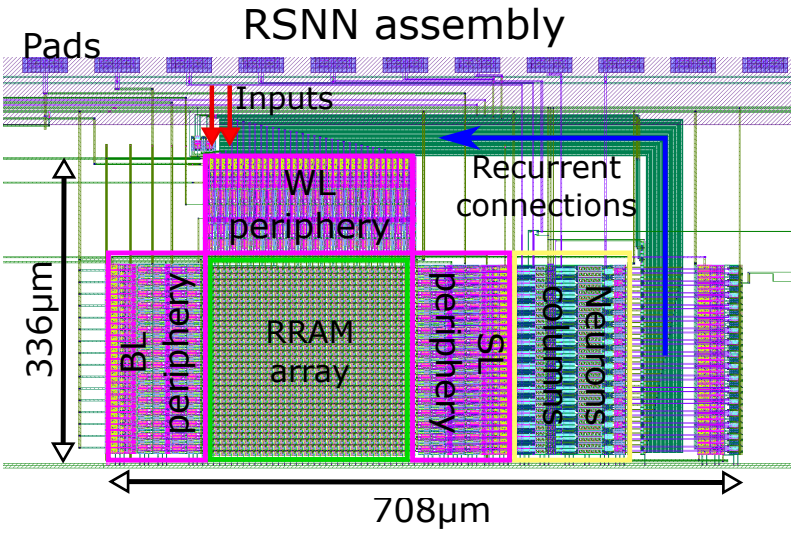


Figure 2.12: Assembly of the RSNN chip. The Periphery of the WL, BL and SL surround the 34x32 array in the middle. Neuron Columns follow to the right and drive the Recurrent Connection realized with metal wires.

When source lines are not selected for programming, they can be operated in "Inference" mode, meaning that the array is connected to the neuron columns and the chip works as a RSNN. A simple MUX is placed between the SL periphery and the Neuron Column to switch the connection between the two blocks and the RRAM array's SLs, differentiating between "programming" and "inference" mode. The output of the neuron columns are the voltage spikes which are fed back recursively to other neurons. This physical connection is made by routing outputs from neuron columns to the word lines of the RRAM array, by means of the metal wires highlighted by the blue arrow in Fig. 2.12. Such routing is not the most compact in terms of footprint area, but it was selected for its simplicity.
The chip also features a small block which selects 1 of the 32 neuron columns and buffers 3 states to as many output pads. The 3 states of the neuron column are the DPI synapse's voltage, the LIF neuron's membrane voltage and the output spikes. This is fundamental to retrieve the output of the RSNN network. To do this, a 3-way MUX is stacked 32 times (one per neuron column) with 32 selectors. The selectors are driven by a scan chain similar to that utilized in the array peripheral circuit. The block is adjacent to the neuron columns.
Overall, the chip has a footprint area of [708x336]μm^2.

Testing RRAM-based fully analog systems is complicated because asynchronous analog signals have to be transmitted in and from the chip. Furthermore, the limited number of pads in the test structures (25 in this case) imposes limits to what can be read during

the functioning of the circuit. For this reason, the circuit has been rescaled to different sizes, from a 1x, to 8x up to a 36x32 RSNN. In the AxB nomenclature, A represents the word line count and B the bit/source line count. B is also the number of recurrent neurons. Smaller circuits will permit a better understanding of the neuron column circuit. Larger arrays are dedicated to perform inference on the RSNN they implement. To recall the NHC procedure, this circuit's test will feature a first phase of characterization of the neuron columns (DPI synapse and LIF neurons). Then, a second phase where the network is trained off-line with the surrogate gradient technique. Lastly, the weights are transferred to the RRAM array and the input are fed. The expected result is that the chip will correctly classify the ECG dataset. This dataset is chosen as it only features 2 (or 4) input channels.

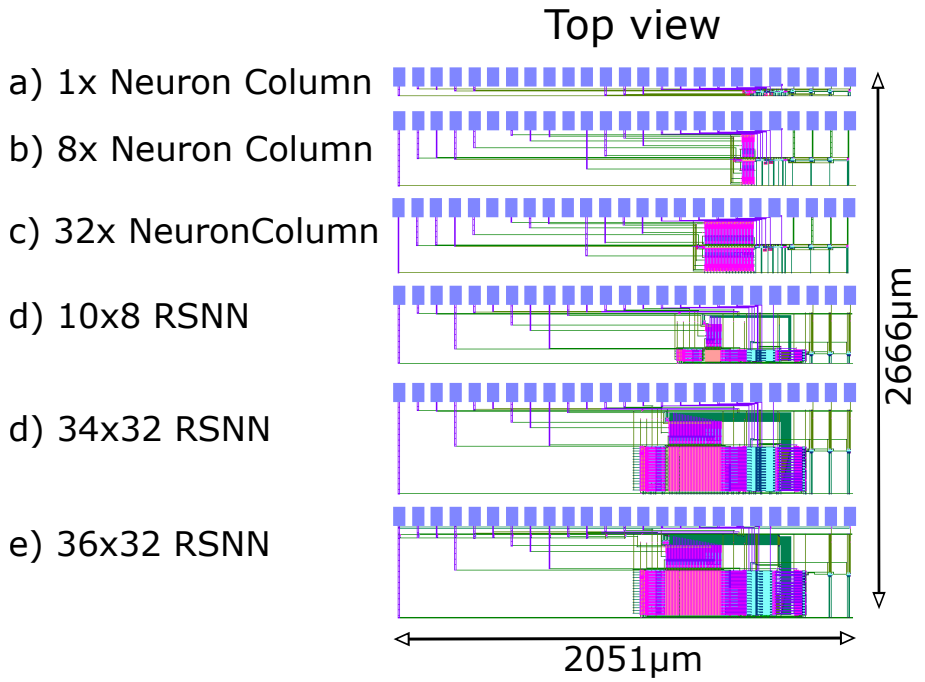


Figure 2.13: Top view of the submitted mask. This includes 5 circuits with different scales of the RSNN circuit design described in the previous section, ranging from a 1x neuron column to a 36z32 RSNN.

Limits of this approach Analog electronics and RRAM-based computations are extremely interesting technologies with a lot of potential, but are not perfect. Such approaches present fundamental technical difficulties which set some limitations. At the same time, the design choice made to build the chip are far from optimal and the RSNN design presented in this section has a lot of room for improvement. The main limitations of this approach are the following:

- IR drop

- Capacitive effects
- Large footprint area
- High RRAM conductance

IR drop is a problem affecting large arrays in particular, and manifests in a loss of voltage along a long line in an array. If, for example, one needs to program a device at the end of a line in a RRAM array with a voltage V_{prog}, the effective voltage across of the device will be inferior. This is because of the resistive losses across the array's metal wire. IR drop is a well-known problem in integrated circuits and in RRAM array design [155]. The limitation due to IR drop is that the array cannot be scaled indefinitely.
Capacitive effects are also related to long metal lines which need to be driven with large current to transmit voltage pulses. In the presented RSNN, critically long wires are present both in the array (word lines) and as the recurrent connections. Capacitive effect limits the minimum voltage pulse-width of spikes, which need to be long enough to be transmitted correctly. However, long pulses increment power consumption.
Footprint area is a key aspect of integrated circuits as it dramatically impact their costs. The presented RSNN features LIF neurons and DPI synapses which are based around bulky capacitors. The capacitors have to be big enough to assure large time constant of the membrane and synaptic potentials. Long time constant are important for the performance of the RSNN as the temporal dynamics of the network has to be comparable to that of the sensory input signals. Possible solution of such problem might rely in the utilization of high-k dielectrics or special devices featuring capacitive effects [156, 157].
RRAM's conductance is an important factor impacting power consumption. A lower conductance range would reduce the current read from the device and the power consumption in turn. Energy associated to reading RRAM is instead an important factor in the RSNN design, as seen in Figure 2.6. Utilizing different devices with lower conductance range is a possible improvement of the design. Possible alternatives to RRAMs are PCM, FTJ [80] and FeRAMs [78].

2.5 Scaling up with the Mosaic concept

Analog electronics, RRAM-based implementations of neural network offer great benefits regarding the energy consumption, but also come with the mentioned limitations. Despite event-based models don't overcharge the column current in RRAM arrays (Fig. 2.1), scaling networks to large sizes faces the challenges due to IR drop and detrimental capacitive effects. The question then is: *is a single RRAM array an effective implementation of a large RSNN?* At the same time, one aims at improving the efficiency of computation and a questions arises: *is the fully-connected architecture in RSNN the most effective computational method?*
Both question find an answer in a novel modular architecture called Mosaic, a novel approach for building RSNNs based on a systolic array of memristor crossbars. The Mosaic concept is based on breaking a single larger RRAM array into smaller tiles that distribute computation, overcoming the issues of IR drop along large word and course lines. Furthermore, the Mosaic architecture introduces special tiles dedicated to route spikes across the large systolic array. While the tiles dedicated to computation are called "Neuron Tiles", the ones dedicate to routing are defined as "Routing Tiles". In the routing tiles, RRAM are used in a binary fashion to either transmit or block spikes across the chip. These router mitigate the problem of the detrimental capacitive effects of long metal wires for communication on the chip. Also, as RRAMs are plastic element, the Mosaic is a re-configurable architecture onto which the connectivity of the underlying neural network can be mapped. Thanks to its topology, Mosaic is particularly well-suited for the implementation of small-world graphical models, with dense local and sparse global connectivity - found extensively in the brain. We mathematically show how Mosaic exploits this connectivity to reduce the memory footprint, an advantage which becomes greater as the neural network size scales up.
The proposed architecture is tested to solve Electrocardiogram anomaly detection and spoken digit recognition tasks. The advantage of Mosaic is evident in the total energy required for communication relative to other approaches. Mosaic promises to open up a new approach to designing neuromorphic hardware based on graph-theoretic principles with less memory and energy.

2.5.1 The opportunities of small-world graphs computation

Graphs are omnipresent data structures which capture interactions (i.e., edges) between multiple units (i.e., nodes). They are the backbone of many computational systems that represent relational information between their interacting entities [159]. Graphs can be used to study and represent both biological and artificial neural networks, where neurons correspond to the nodes of a graph and the connections between them (i.e., weights or synapses) correspond to edges. Biological nervous systems, shaped over millions of years of evolution, have developed many computational principles that can be captured using graphical networks. Therefore, building computing architectures based on the

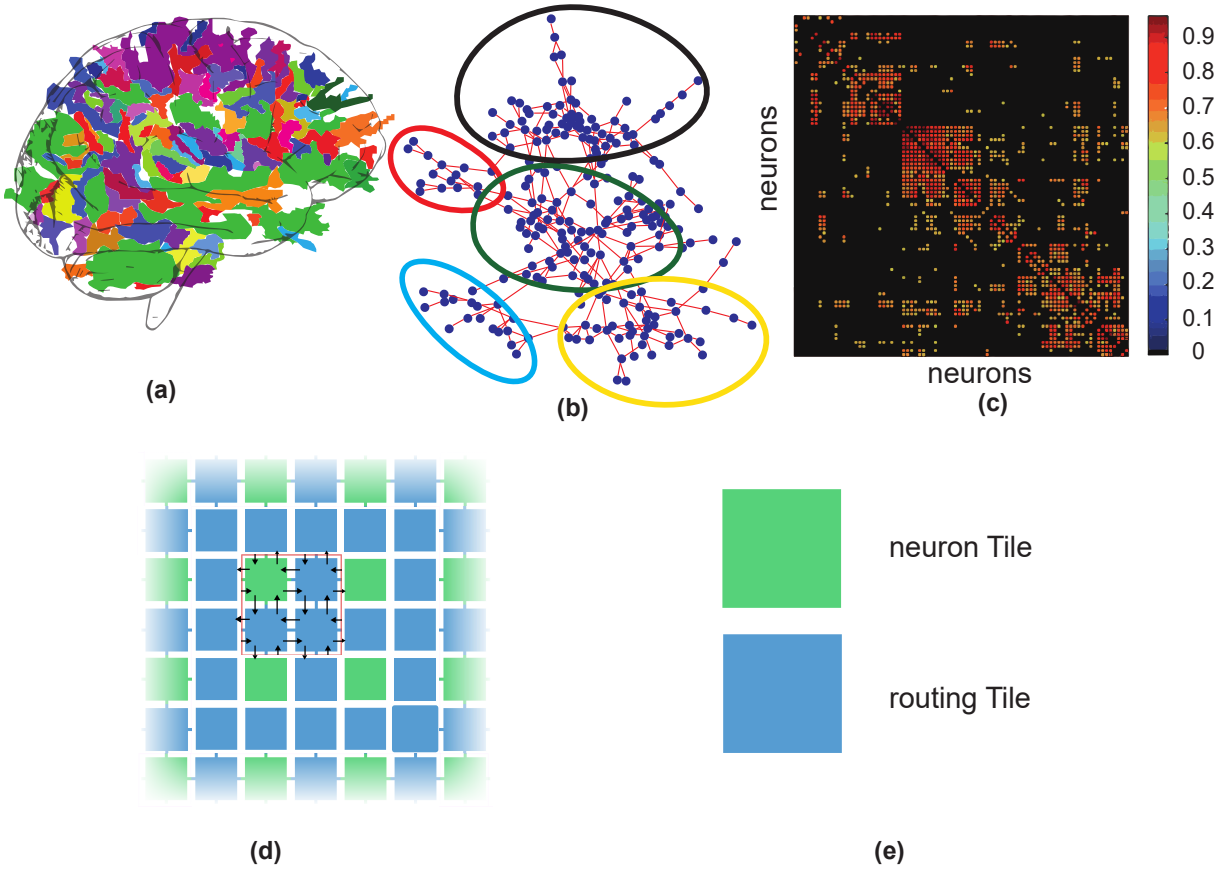


Figure 2.14: Small-world graphs in biological and graphical neural networks. (a) Depiction of small-worldness in the brain with highly clustered neighboring regions highlighted with the same color. (b) An example network model characteristic of a small-world graph. Five local clusters of nodes connect densely with each other and are interconnected with a sparse set of hub-like nodes. (c) (adapted from [158]). The functional connectivity matrix based on data from human functional Magnetic Resonance Imaging showing the properties of a small-world graph. The rows and columns represent neuron indices. The diagonal region of the matrix contains the strongest connectivity which represent the connections between the neighboring neurons. The off-diagonal elements are not connected. (d) High level representation of the Mosaic architecture. Small tiles are disposed into a mesh, where communication occurs between neighbouring tiles only, as shown by the inset of the image . (e) The Mosaic architecture features types of "tiles": computation is performed in neuron tiles, while routing is performed by routing tiles.

same organizational principles is a promising path towards realizing powerful artificially intelligent systems.

One such important organizing principle is “small worldness” which is found extensively in empirical studies of structural and functional biological neural networks [160, 161]

(Fig. 2.14a). In such a structure, short paths connecting neighboring nodes (neurons) are more common than long-range connections, which are sparse (Fig. 2.14b). The mix of dense local and sparse distal connectivity gives rise to efficient global coordination and information flow based on local interactions [162]. A connectivity matrix of an example small-world graph is plotted in Fig. 2.14c. It is characterized by heavy connectivity along the matrix diagonal, with increasingly fewer connections between the further off-diagonal neuron pairs.

Therefore, going beyond traditional RRAM crossbar architectures has been receiving increasing attention [163, 164, 165, 166]. To implement artificial spiking small-world graphs more efficiently, architectures with more local connectivity is required which results in a better utilization of the allocated memory resources. For example, the CMOL Crossnet architecture [167] permits local neuron connectivity through small tilted distributed RRAM crossbars. However, the required tilt makes the integration challenging. Moreover, long range global connections can only be obtained through 3D stacking of successive layers of memory devices.

To solve these challenges in a practical fashion, we propose and experimentally demonstrate a new re-configurable neuromorphic computing architecture called the "Mosaic" (Fig. 2.14(d)). The Mosaic is a two-dimensional systolic matrix of distributed "tiles", each based on a small crossbar of RRAM, that can serve either as analog spiking or spike routing elements. Effectively, the Mosaic dices up one large crossbar into numerous smaller tiles with different functions (Fig. 2.14(e)). Importantly, the Mosaic uses RRAM not only to store synaptic weights and carry out neural processing, but also to define the routing patterns linking up neighboring tiles. The Mosaic is configurable on-the-fly and can be implemented with standard CMOS technology integrated with a single layer of RRAM. Moreover, the Mosaic introduces a novel routing approach different from the conventional AER scheme in SNN hardware [168, 29] without the need for storing each neuron's connectivity information in local memories that draw static power and can consume a large chip area.

2.5.2 The Mosaic architecture

The Mosaic architecture is illustrated in Fig. 2.14d as an array of tiles which are distributed in a two-dimensional systolic fashion. Each of the tiles consist of a small memristor crossbar which can receive and transmit spikes to and from their neighboring tiles to the North (N), South (S), East (E) and West (W) directions. The green squares represent "neuron tiles" and correspond to small crossbars (Fig. 2.14e) that store the synaptic weights of several LIF neurons. These neurons are implemented using analog circuits and are located at the termination of each row, emitting voltage spikes at their outputs [23]. Effectively, when recurrent connections are enabled, neuron tile work as small RSNNs. The spikes are communicated between neuron tiles through a mesh of blue squares which represent "routing tiles". Routing tiles are also small crossbars that determine the connectivity patterns

between neuron tiles. The state of each device in a routing tile crossbar determines the output direction (i.e., N, S, E, W) towards which its input spike propagates, i.e. steering it towards its intended target neuron elsewhere in the Mosaic. Together, the two tiles give rise to a continuous *mosaic* of neuromorphic computation and memory for realizing spiking small-world neural networks.

An example of small-world neural network topology, obtained by randomly programming memristors in a computer model of the Mosaic is shown in Fig. 2.15a. The resulting graph exhibits an intriguing set of connection patterns that reflect those found in many of the small-world graphical motifs observed in animal nervous systems. For example, central 'hub-like' neurons with connections to numerous nodes, reciprocal connections between pairs of nodes reminiscent of winner-take-all mechanisms, and a number of heavily connected local neural clusters [161]. If desired, these graph properties could be adapted on-the-fly by the re-programming the RRAM states in the two tile types. For example, a set of desired small-world graph properties can be achieved by randomly programming the RRAM devices into their HCS with a certain probability. Random programming can for example be achieved elegantly by simply modulating RRAM programming voltages [154].

For Mosaic-based small-world graphs, we estimate the required number of memory devices (synaptic weight and routing weight) as a function of the total number of neurons in a network, through a mathematical derivation. Fig. 2.15b plots the memory footprint as a function of the number of neurons in each tiles for different network sizes. Horizontal dashed lines show the number of memory elements using one large crossbar for each network size, as has previously been used for RNN implementations [169]. The cross-over points, at which the Mosaic memory footprint becomes favorable, are denoted with a cross. While for smaller network sizes (here 128 neurons) no memory reduction is observed compared to a single large array, the memory saving becomes increasingly important as the network is scaled. For example, given a network of 1024 neurons and 4 neurons per neuron tile, the Mosaic requires almost one order of magnitude fewer memory devices than a single crossbar.

Calculation of memory footprint We calculate the Mosaic architecture's Memory Footprint (MF) in comparison to a large crossbar array, in building small world graphical models.

To evaluate the MF for one large crossbar array, the total number of devices required to implement any possible connections between neurons can be counted - allowing for any SRNN to be mapped onto the system. Setting N to be the number of neurons in the system, the total possible number of connections in the graph is $MF_{ref} = N^2$.

For the Mosaic architecture, the number of RRAM cells (i.e., the MF) is equal to the number of devices in all the neuron tiles and routing tiles: $MF_{mosaic} = MF_{NeuronTiles} + MF_{RoutingTiles}$.

Considering each neuron tile with k neurons, each neuron tile contributes to $4 \times k^2$ devices (where the factor of 4 accounts for the four possible directions to which each tile can connect). Evenly dividing the N total number of neurons in each neuron tile gives rise to $T = ceil(N/k)$ required neuron tiles. This brings the total number of devices attributed to the neuron tile to $T \times 4 \times k^2$. The number of routing tiles which connects all the neuron tiles depends on the geometry of the Mosaic systolic array. Here, we assume neuron tiles assembled in a square, each with a routing tile on each side. We consider R to be the number of routing tiles with $4k^2$ devices in each. This brings the total number of devices related to routing tiles up to $MF_{RoutingTiles} = R \times (4k)^2$. The problem can then be re-written as a function of the geometry. Considering Fig.2.14d, let i be an integer and $(2i+1)^2$ the total number of tiles. The number of neuron tiles can be written as $T = (i+1)^2$, as we consider the case where neuron tiles form the outer ring of tiles. As a consequence, the number of routing tiles is $R = (2i+1)^2 - (i+1)^2$. Substituting such values in the previous evaluations of $MF_{NeuronTiles} + MF_{RoutingTiles}$ and remembering that $k < N \times T$, we can impose that $MF_{Mosaic} = MF_{NeuronTiles} + MF_{RoutingTiles} < MF_{MF_{ref}}$. This results in the following expression:

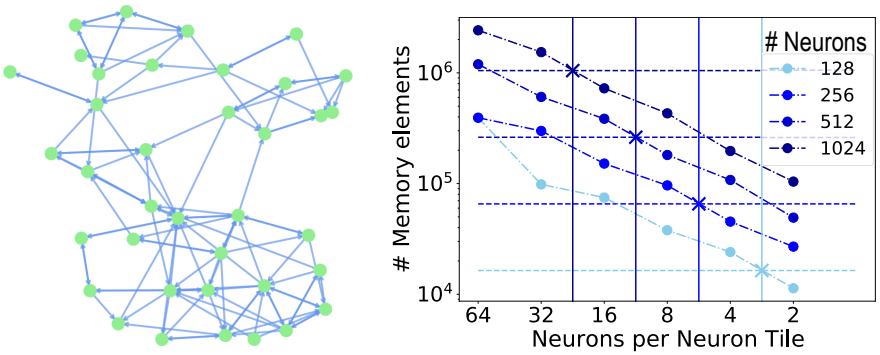
$$MF_{Mosaic} = MF_{NeuronTiles} + MF_{RoutingTiles} < MF_{reference} \tag{2.2}$$

$$(i+1)^2 4 \times k^2 + [(2i+1)^2 - (i+1)^2]((4k)^2) < (k(i+1)^2)^2 \tag{2.3}$$

This expression can then be evaluated for i, given a network size, giving rise to the relationships as plotted in Fig.2.15b.

Neuron tile circuits: small worlds

Each neuron tile in the Mosaic is composed of multiple "neuron columns"; a circuit that models a LIF neuron and its synapses. To realize a small RSNN, neuron columns are agglomerated into a 'tile'. This is done through stacking consecutive columns side-by-side and connecting their gates row-wise to common input lines (i.e., a crossbar architecture). A simple neuron tile, composed of only two neuron columns receiving two inputs, is shown in Fig. 2.16a. The top two rows of the crossbar represent the neurons' synaptic weights corresponding to external inputs, while the bottom two represent those of the recurrent connections between neurons within the tile. Following a systolic organization [170], each input or output spike can enter from, and exit towards, the neighboring N, S, E, W tiles. We mapped a simple network topology onto a fabricated neuron tile circuit depicted in Fig. 2.16a. Two devices highlighted in bold black were programmed to be in their HCS while the gray shaded ones were programmed in their LCS. We then applied a train of input voltage spikes to $V_{in} < 0 >$. The experimental measurements are plotted in Fig. 2.16b whereby the membrane potential of neuron 0 is observed to periodically increase upon the arrival of each pulse. After the 6^{th} input pulse, V_{mem} exceeds the threshold V_{th}, and the circuit generates an output spike. Because of the recurrent connection between the two neurons defined in the neuron tile, the membrane of neuron 1 integrates an excitatory



(a) (b)

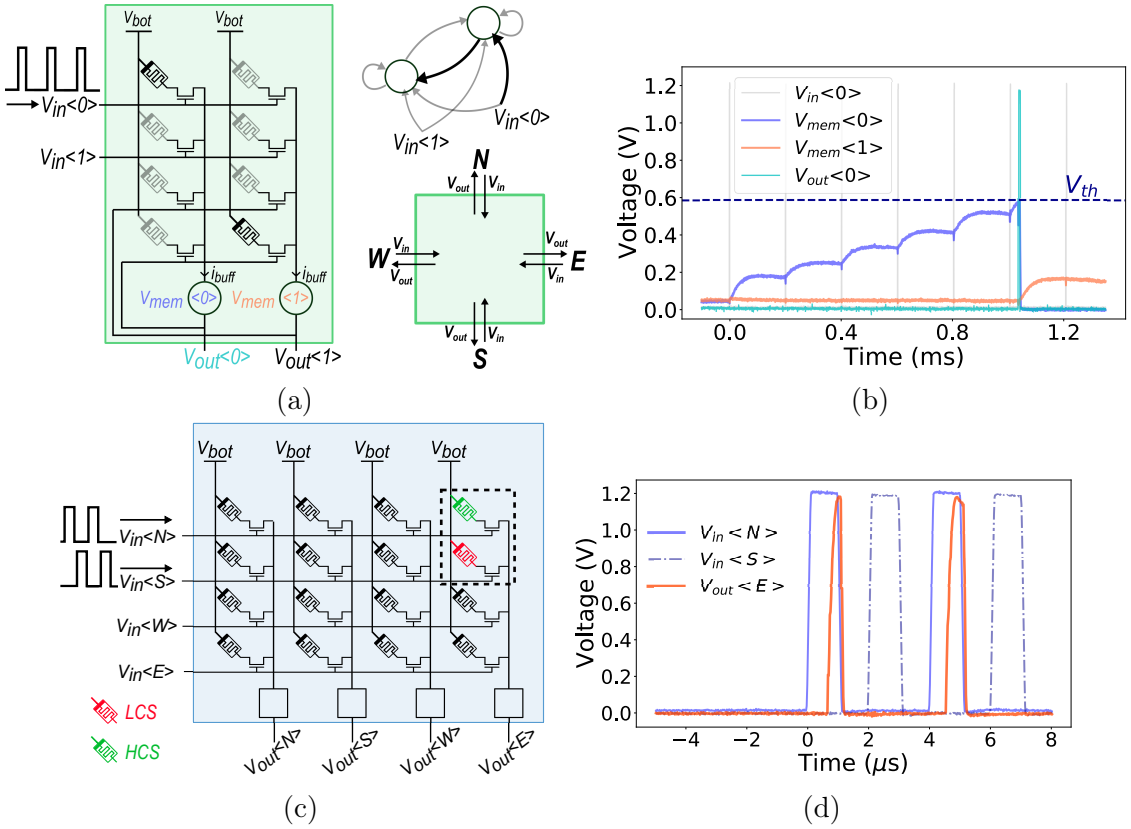
Figure 2.15: (a) An example graph resulting from the random programming of devices in each of the tiles in the Mosaic pictured in part. The green circles correspond to neurons which exist in the neuron tiles and the blue edges are defined by the resulting paths that are formed between neuron tiles through the routing tiles. (b) Plot showing the required bits of memory for different number of total neurons in a network model depending on the size of the neuron tile. The number of bits of memory is referred to as resistive memory devices programmed in a binary fashion. The horizontal dashed line indicates the number of required memory bits using a fully-connected RRAM crossbar array for different network sizes. The cross (X) illustrates the cross-over point beyond which Mosaic approach becomes favorable.

post-synaptic potential at the same instant (shown in orange). Neuron 0 then enters a temporary refractory period, during which it does not integrate incoming spikes.

Routing tile circuits: connecting small-worlds

A routing tile circuit is shown in Fig. 2.16c. It acts as a flexible means of configuring how spikes emitted from neuron tiles propagate locally between small-worlds. The functional principles of the routing tile circuits are similar to the neuron tiles. The principal difference is the replacement of the biological synapse and neuron circuit models with a simple current comparator circuit. On the arrival of a spike on the column, it compares the device read current to a reference. If it is larger than this reference, it generates an output spike. Otherwise the output remains at zero. Therefore, the state of the device serves to either pass or block input spikes: in Fig. 2.16c, each device determines whether input spikes arriving from different input ports (N, S, W, E) are propagated, or not, to each output port.

Using a fabricated routing tile circuit, we demonstrate its functionality experimentally. Two devices (colored in green and red in Fig. 2.16c) were programmed in HCS and LCS respectively. The other devices were left in the pristine state. This has the effect of allowing incoming pulses from N to propagate out to E, while blocking pulses coming from S direction. A pair of pulses were applied to N and S input ports of the fabricated



(a) (b) (c) (d)

Figure 2.16: **Experimental measurements of the fabricated tile circuits.** (a) Depiction of a neuron tile. Devices are colored in black or gray to indicate respectively whether they were in HCS or LCS during our experimental results plotted in part b. Colored voltage labels and input voltage pulses are also in reference to plots in part b. The input and output voltage pulses can come from or be propagated to the neighboring tiles to the North (N), South (S), East (E) and West (W). (b) Voltage traces measured from a fabricated implementation of the neuron tile circuit in a. Due to an input pulse train (gray pulses) at $V_{in} < 0 >$ the membrane of the zeroth neuron column in the tile integrates an increasing amount of voltage (purple trace) until, after six pulses, the neuron fires (light blue trace). As a result of the feedback connection to the other neuron column, neuron 1 also exhibits an increase in its membrane voltage. (c) Circuit schematic of a routing tile. Two devices colored green and red denote respectively the devices programmed in the HCS and LCS in the experiment of part (b). Rectangular pulse waveforms depicted on the left-hand side indicate where the input voltage pulses were applied during this experiment. (d) Experimental results from a fabricated version of the routing tile shown in part (c). Continuous and dashed blue traces show the waveforms applied to the N and S inputs while the orange trace shows the response of the output towards the E port. The E output port follows the N input resulting from the device programmed into the HCS in part (a).

circuit, plotted respectively in solid and dashed blue lines in Fig. 2.16d. While the E output port remains at zero due to the incoming pulses from the S input port, it switches to a high voltage as a result of incoming pulses from the N input port. This output pulse propagates on-wards to the next tile. Note that in Fig. 2.16d the output spike does not appear as rectangular due to the large capacitive load of the probe station. To allow for greater configurability, more channels per direction can be used in the routing tiles.

Calculation of routing energy In state-of-the-art event-based neuromorphic chips, the information is communicated through the AER scheme [168]. Whenever a spiking neuron in a chip (or module) generates a spike, its "address" (or any given ID) is written on a high-speed digital bus and sent to the receiving neuron(s) in one (or more) receiver module(s). In the Mosaic structure, we have distributed the routing information in a two-dimensional matrix along with the computing units. To compare the routing energy and latency of Mosaic with the AER systems, we have calculated the energy per spike routing in the best and worst case scenarios in both systems.
In Mosaic, routing is performed by Routing Tiles, and an input spike is passed onto a neighbouring Neuron Tile by activating the circuit in Fig. 2.16c. To assess the energy efficiency of such circuit, a SPICE simulation is performed. The simulation involved an input spike activating a RRAM set at 10 $k\Omega$ and activating the output comparator, buffering the spike at the output. The input spike is assumed as a voltage pulse of 10 ns of duration and 1.2 V of voltage. As intuitive, the duration of the pulse is a critical design parameter affecting the energy consumption. The pulse-width of the spike was chose to satisfy the requirement of the correct working of the circuit under process variation: such pulse-width allows correct the functioning of the routing tile circuit 97% of the times, as verified in Monte-Carlo simulations. The resulting energy per routing operation is as low as 60 pJ. For AER-based systems, we are using the energy and latency numbers reported in Dynap-SE, as one of the most recent and optimized AER routing schemes [29]. It is a multi-core neuromorphic circuit comprising four cores; each core includes 256 neurons. It has a hierarchical asynchronous routing, combining a source-based routing mesh architecture with a destination-based hierarchical tree routing method. SRAM cells store the routing structure in the tree and the CAM cells store the tag of the source address to which each neuron is connected.

Therefore, once a spike is generated, the least energy consumption happens in a scenario where the events should be routed locally, and thus 256 10-bit CAM cells are accessed. In the worst case, events have to travel from the first-level router to the higher levels and thus the energy of reading SRAM cells are added. Therefore, the energy of routing one spike in Dynap-SE can be calculated by the following equation:

$$E_{total} = E_{Spike} + E_{Pulse} + E_{En} + E_{BC} + RT.E_{RT} \tag{2.4}$$

Where E_{Spike} is the energy to generate one spike, E_{Pulse} is the energy of the pulse extender

circuit, E_{en} is the energy to encode one spike and append destination, E_{BC} is the energy to broadcast the event to the same core, RT is 1 if the spike has to be routed to other cores, otherwise zero, and E_{RT} is the energy to route the events to other cores. If $RT = 0$, total energy to route the event to the core sums up to 7.68 nJ. In case of the event routing to other cores, multiples of 360 pJ should be added to the energy consumption (energy required for reading SRAM at each hierarchical router level.

2.5.3 Application to real-time sensory processing

The small-world connectivity and the spiking nature of the Mosaic architecture imposes spatial and temporal sparsity on the network. Additionally, the use of RRAM devices as synaptic weight elements imposes a severe limitation on the precision of the stored weights relative to an equivalent software model. To evaluate the effect of the sparsity in combination with our fabricated circuit characteristics on the Mosaic performance, we benchmark it on two real-world sensory applications against software solutions. Specifically, we compare the prediction accuracy of an ideal software RSNN [171, 110, 112] to that of a small-world RSNN implemented with the Mosaic approach, (i) to detect arrhythmic heartbeats within ECG signals [150], and (ii) to classify spoken digits [149].

ECG anomaly detection First, we encode the continuous ECG time-series into trains of spikes using a delta-modulation technique, which describes the relative changes in signal magnitude [172, 173]. These spikes are fed as input into the Mosaic small-world RSNN. As outputs, we designated two sub-populations of neurons within two pairs of the Mosaic's neuron tiles. Elevated spiking activity in either sub-population denotes a normal heart beat (black), or an anomalous one (red) (Fig. 2.17a). Relative to other approaches (e.g. in reservoir computing [166, 174, 175]), this scheme avoids the need for an output feed-forward layer, simplifying the read-out to monitor the state of the output neurons.

We train the RSNN in an ex-situ fashion [135], using BPTT [176] with surrogate gradient approximations of the derivative of a LIF neuron activation function [111]. We then transferred the resulting floating-point precision weights to the low-precision conductance states of memristors in an experimental crossbar using a closed-loop iterative programming algorithm [129]. The resulting conductances, corresponding to an equivalent large-scale implementation of the Mosaic, are then read during a mixed hardware-software co-simulation of the system.

An example of the resulting spike trains produced in the Mosaic, due to an ECG time-series of the arrhythmic heartbeat plotted in Fig. 2.17a, is shown in Fig. 2.17b. The activity of the neurons in each predictive sub-population are bounded within red and black horizontal dashed lines. The neurons in the red sub-population fire more frequently than those in the black sub-population, here correctly identifying the heartbeat as arrhythmic.

The accuracy over the test set for 100 iterations of training, transfer and test is plotted in Fig. 2.17c using a boxplot. The median detection accuracy of the Mosaic is 96.9%. To gauge the effect of the spatial and temporal sparsity in the Mosaic architecture, we compare it to two networks: one network without spatial sparsity, but including temporal sparsity (Software SNN), and the other one without any sparsity (Software RNN). While compared to the Software SNN, with a median accuracy of 97.0%, the Mosaic suffers a negligible difference in average accuracy, it outperforms the Software RNN (with the median accuracy of 96.1%). This result is consistent with other observations whereby RSNN have outperformed non-spiking equivalents [177]. This illustrates that, not only the imposed sparsity of the Mosaic does not have a negative effect on the accuracy, but that the model is also robust to a severe degradation in the precision of the weights for this task.

Spoken digit classification We next apply the Mosaic to the more challenging Spiking Heidelberg Dataset (SHD) [149]. The Mosaic-based model was trained using the same NHC approach as in the ECG task, and we compare its performance to a software-based RSNN. To classify the SHD dataset, a SNN network featured 144 neurons, 20 of which representing the outputs, distributed across 36 neuron riles. Connections between neurons in the Mosaic are enabled given the SET probability for the neuron and routing tile devices, assuring sparse connectivity. From the Mosaic's neurons, 20 output neurons are chosen in uniformly spaced fashion, and are isolated from the rest of the neurons by disabling their recurrent connections to other neuron tiles. The readout of the output is conveniently performed by monitoring the state of the 20 output neurons in the Mosaic architecture, without requiring external computation. The results are reported in Fig. 2.17d. Mosaic suffers from a slight loss of accuracy compared to the unconstrained software model [149]. The difference can be attributed to the fact that the Mosaic's output neurons are chosen inside the recurrent pool without an explicit output layer. Representing the weights in the Mosaic architecture with 8-levels RRAM values results in a slight drop of performance, due to the reduced precision of the weights. This is compensated by the advantages in energy consumption and density that the RRAM devices bring to the Mosaic architecture.

Efficiency of Mosaic compared to a fully-connected RSNN Fig. 2.17e and Fig. 2.17f respectively depict the required memory and energy footprint of Mosaic compared to a fully-connected RSNN for ECG and SHD tasks. For the smaller size network required for the ECG task, Mosaic requires slightly higher memory and energy consumption. However, as the size of the network increases, i.e. in the case of SHD task, Mosaic shows significant benefits, i.e. about 66% less memory and energy footprint. This is in agreement with the mathematical estimations of the memory (and thus power) consumption in Fig. 2.15b.

Table 2.2: Comparison of routing performance

Chip	True North [178]	SpiNNaker [179]	Neurogrid [180]	Dynap-SE [29]	Loihi [25]	Crossnet [181]	Mosaic
Techn.	28 nm	130 nm	180 nm	180 nm	14 nm	45 nm°	130 nm
Routing	on-chip	on-chip	on/off-chip	on-chip	on-chip	on-chip	on-chip
Routing energy	2.3 pJ @ 0.775 V	4 nJ	18.8 nJ	360 pJ @1.8 V	4 pJ @ 0.75V	0.5 pJ	60 pJ @1.2V
Energy Scaled to 130 nm	49.5 pJ	4 nJ	9.74 nJ	190 pJ	344 pJ	18 pJ	60 pJ
Routing latency	6.25 ns	200 ps	20 ns	40 ns	6.5 ns	N/A	29 ns
Latency Scaled to 130 nm	29 ns	200 ps	14.4 ns	28.88 ns	60.35 ns	N/A	29 ns

° For a 64-bit crossbar with $F_{nano} = 10\,nm$.

Comparison to other approaches

We compare the efficiency of the Mosaic architecture against existing approaches. We take into account the energy and latency required to route one event to the neighboring computing core. Table 2.2 shows this comparison.

In the table, we have scaled the energy and latency figures of all the platforms to 130 nm technology using general scaling laws[182] for a fair comparison. The scaled energy figures show that although the Mosaic's design has not been optimized for energy efficiency, it outperforms all platforms except for TrueNorth with comparable figures to the Mosaic, and the Crossnet. This efficiency can be attributed to the Mosaic's *in-memory routing* approach resulting in low-energy memory access distributed in the space. This distributed architecture reduces the size of each router, compared to larger centralized routers in other platforms and thus reduces the access energy. Moreover, it avoids the use of content-addressable-memory (CAM) used in many spike routing platforms which are the main source of routing energy. It is worth noting that the energy figures reported for Crossnet are not for routing *per se*, as the architecture does not include any routers, but rather it can send events to any neurons in a local connectivity domain, with their own connectivity domains. The Mosaic's implementation does not require the tilted crossbars in Crossnet, and hence facilitates the fabrication of small-world networks with configurable routers. The Mosaic's latency figure per router is comparable to the average latency of other platforms, which is most often a negligible factor in real-time sensory processing applications.

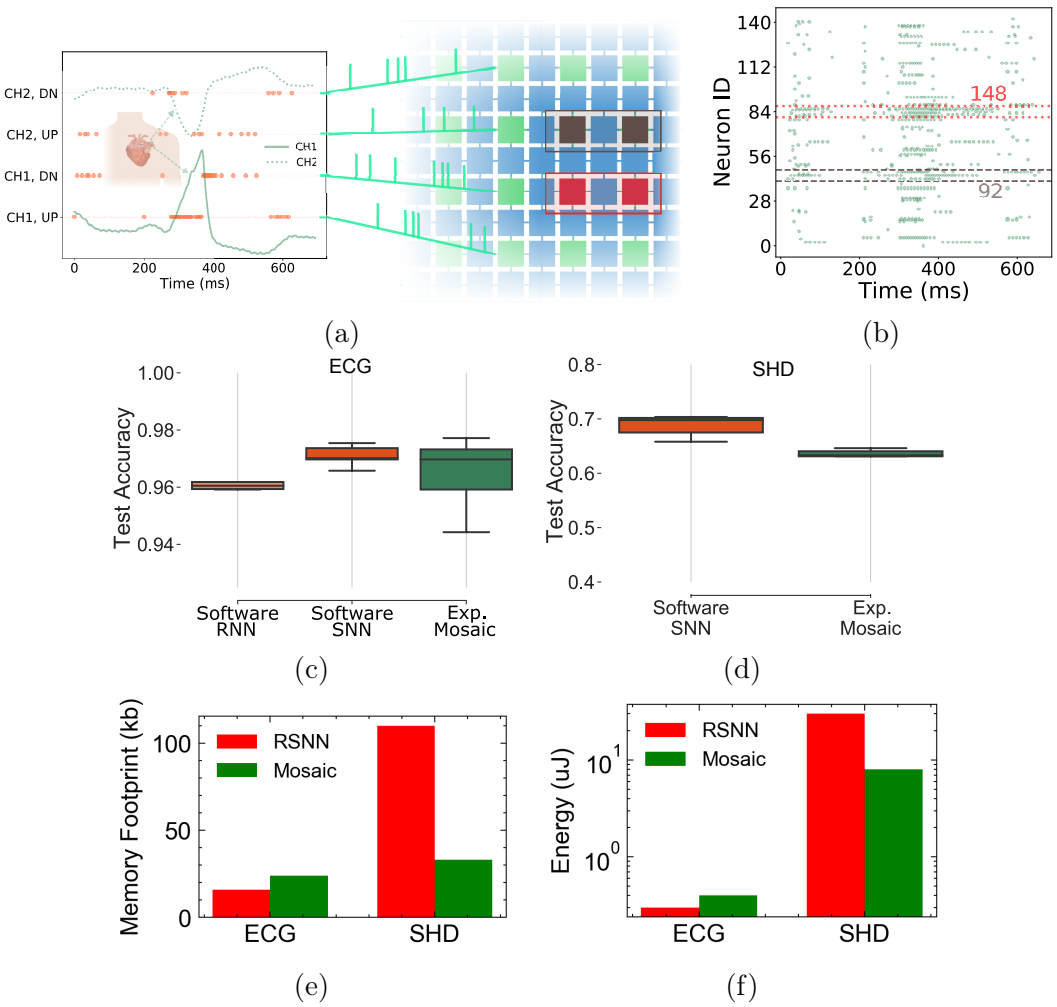


Figure 2.17: **Benchmarking the Mosaic against two tasks of ECG arrhythmia detection and Spoken digit classification (SHD dataset).** (a) A depiction of the ECG classification use case, addressed with the Mosaic architecture. Two-channel ECG waveforms are delta-modulated into four channels. Two groups of two neuron tiles (colored in black and red) are designated as the output neuron populations. (b) An example raster plot of all neurons in the Mosaic in ECG task. Green points indicate the spike times of each neuron. Red and black dashed horizontal lines respectively indicate the anomalous and normal population activity used as the output neurons. (c) A comparison of the accuracy of the ECG anomaly detection task. Boxplots show the accuracy distribution over 10 iterations of a software-based recurrent neural network (left, orange), a software-based spiking neural network (center, red) and the experimental Mosaic model with multi-level resistive memory devices acting as the synapses (right, green). The colored boxes span the upper and lower quartiles of accuracy, while the upper and lower whiskers extend to the maximum and minimum accuracies. (d) Comparison of the accuracy of the SHD task. Boxplots show the accuracy distribution over 10 iterations of a software-based recurrent neural network (red) and the experimental Mosaic model.. (e) Required memory footprint for two tasks of ECG and SHD using a fully connected RSNN, compared to the Mosaic. As the size of the network increases (SHD case), the memory savings for Mosaic is significant (one third of the RSNN). (f) Energy consumption for the two tasks using a fully-connected RSNN compared to the Mosaic architecture.

Chapter 3

Neuromorphic system for object Localization

Neuromorphic engineering is as much about sensing as it is about computation. The previous chapter focused on a computing platform, based on neurons and synapses making use of RRAMs. This computational primitives are now utilized in the context of a neuromorphic system where sensing and processing are co-designed to yield unprecedented energy efficiency. For the first time, advance miniaturized acoustic sensor such as pMUTs are utilized for object localization in a neuromorphic system. Inspiration from biology guides the development of a computational map based on the Owl's auditory cortex. The result is a system - described in this chapter - that leverages advanced technologies such as pMUTs and RRAMs and is based on biologically derived algorithms to optimize the energy efficiency.

3.1 Motivation

We are entering an era of pervasive computing devices where an exponentially increasing number of autonomous devices are being deployed to assist us in our daily lives. These autonomous machines will have to operate continuously, dissipating the lowest possible amount of energy while learning to interpret the data they capture from several sensors in real-time. The first step to this objective is to extract useful and compact information from noisy and often incomplete sensor data [183]. Conventional engineering approaches sample the output sensor signal at high rates, thus generating huge amounts of data, even in the absence of useful input stimuli. Moreover, these approaches require complex techniques to pre-process the noisy data. Biology offers alternative solutions for processing noisy sensory data, using energy-efficient, asynchronous, event (spike)-driven methods [184, 185]. Neuromorphic computing draws inspiration from biological systems to reduce the

computational cost in terms of energy and memory requirements, relative to conventional signal processing techniques [186, 187]. Innovative general-purpose brain-inspired systems that implement spiking neural networks (TrueNorth[33], BrainScaleS[188], DYNAPs[29], Loihi[25], Spinnaker[189]) have recently been demonstrated. These processors offer low-power and low-latency solutions for implementing machine learning tasks, and for modeling cortical circuits. To take full advantage of the energy efficiency of these neuromorphic processors they have to be connected directly to event-driven sensors [190, 191]. However, only few event-driven sensors exist today. Prominent examples are the Dynamic Vision Sensor (DVS) used for motion detection [192, 193, 194] and object tracking [195], the silicon cochlea sensor [196] and the Neuromorphic Auditory Sensor (NAS) [197] used to recognize phonemes, the olfactory sensor [198], and the touch sensor for texture recognition [52, 199].

In this chapter, we developed an event-driven auditory sensor for object localization. Here, for the first time, we present an end-to-end system for object localization which is obtained by coupling the state-of-the-art piezoelectric micro-machined ultrasound transducers (pMUTs) to a neuromorphic resistive memory (RRAM)-based computational map. In-memory computing architectures employing RRAMs, otherwise known as memristors, are a promising solution to reduce energy consumption [134, 200, 201, 74, 130, 129]. Their inherent non-volatility - not requiring active power consumption to store or refresh the information - matches the asynchronous event-driven nature of neuromorphic computation perfectly, resulting in virtually no power consumption when the system is idle. pMUTs are low-cost, miniaturized silicon-based ultrasound sensors able to act as emitters and receivers [202, 203, 204, 205, 206]. To process the signals captured by the embedded sensors, we have taken inspiration from the neuroanatomy of the barn owl [207, 208, 209].
The barn owl *Tyto alba* is known for its exceptional night hunting capabilities made possible by a very efficient auditory localization system. To calculate the position of a prey, the Barn owl's localization system encodes the Time-of-Flight (ToF) of the sound wave coming from the prey when it reaches each of the owl's ears or sound receptors. Given the distance between the ears, the difference between the two ToF measurements (Interaural Time Difference, ITD) makes it possible to compute the azimuthal location of the target analytically. Although biological systems are not adapted to solve algebraic equations, they perform localization tasks very efficiently. The barn owl's nervous system makes use of an array of Coincidence Detector (CD) neurons [207] (*i.e.* neurons able to detect temporal correlations between spikes propagating down converging excitory terminals) [210, 211] organized into a computational map to solve the localization task.

Previous studies have shown that complementary metal-oxide-semiconductor (CMOS)-based neuromorphic hardware inspired by the inferior colliculus ('auditory cortex') of barn owl constitute an efficient way to compute the position from the ITD [191, 212, 213, 214]. However, the potential of a full neuromorphic system that couples auditory signals to the neuromorphic computational map has not yet been proven. The main challenge is the intrinsic variability of analog CMOS circuits, affecting the coincidence detection precision.

In this work we propose to exploit the ability of RRAMs to change their conductance value in a non-volatile manner to counteract the variability in analog circuits.

We implemented an experimental system consisting of a single emitting pMUT membrane working at 111.9 kHz, two reception pMUT membranes (sensors) that emulate the barn owl's ears, and a neuromorphic computational map fabricated by co-integrating a 130 nm CMOS processor with hafnium dioxide RRAM devices. We experimentally characterized the pMUT sensory system and the RRAM-based ITD computational map to validate our localization system and to estimate its resolution. We compared our approach to a microcontroller performing the same localization task using either conventional beamforming or neuromorphic techniques. We find that the proposed neuromorphic system achieves a reduction in power consumption of five and four orders of magnitude with respect to the two microcontroller-based solutions.

3.1.1 Biological background

One of the most striking examples of precise and efficient object localization systems can be found in barn owls [216, 207, 209]. At dusk and dawn, barn owls (*Tyto Alba*) actively search for small prey such as voles or mice relying mostly on passive listening. These auditory specialists can locate auditory cues incoming from their prey with astonishing accuracy (about 2°) [207], as shown in Fig. 3.1a. Barn owls infer the localization of a sound source in the azimuthal (horizontal) plane from the difference between the ToF incoming from the source on the two ears (ITD). The ITD computation mechanism has been postulated by Jeffress [217, 218], it relies on neural geometry and requires two key ingredients: axons, neuron's nerve fibers, that act as delay lines, and an array of Coincidence Detector neurons organized into a computational map, as depicted in Fig. 3.1b. The sound reaches the ears with an azimuth-dependent time delay (Interaural Time Difference, ITD). In each ear, the sound is then converted into a spike pattern. Axons from the left and right ears act as delay lines and converge at CD neurons. In theory, only one neuron of the array of coincidence neurons will receive simultaneous inputs (where the delay is compensated exactly), and will fire maximally (neighboring cells will fire too, but at a lower rate). This concept is summarized in Fig. 3.1c: for example, if the sound originates from the right, a coincidence will occur when the input signal from the right ear travels a longer path than from the left ear by an amount compensating the ITD, *e.g.* at coincidence neuron 2. In other words, each CD responds to a specific ITD (also called Best Delay) because of axonal delays. In this way, the brain transforms temporal information into spatial information. Anatomical evidence has been found for this mechanism [209, 219]. There are phase-locked neurons of the Nucleus Magnocellularis who preserve the temporal information of the input sound: as their name indicates, they fire at a specific phase of the signal. The coincidence detector neurons of the Jeffress model can be found in the Nucleus Laminaris. They receive input from neurons of the Nucleus Magnocellularis, whose axons serve as delay lines. The amount of delay provided by delay lines may be explained by axonal lengths, but also by differential myelination patterns, changing the conduction speeds.

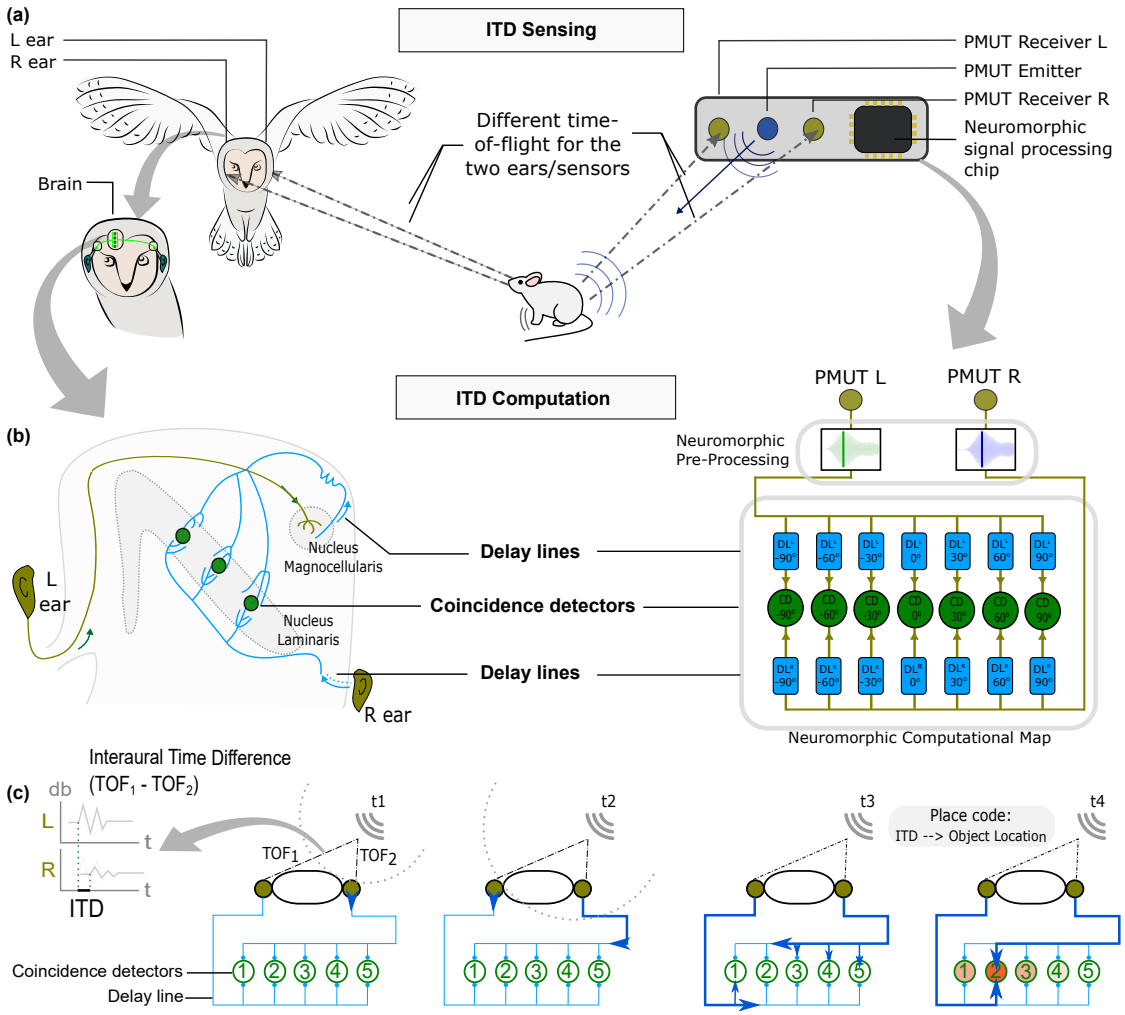


Figure 3.1: Object localization system in barn owls and the proposed bio-inspired technology. (a) The barn owl receives a sound wave from a target, a moving prey in this case. The Time-of-Flight (ToF) of the soundwave at each ear is different (unless the prey is exactly in front of the owl). The dash-dotted lines represent the path of the sound wave towards the barn owl's ears. Based on the difference in the two soundwave path lengths and the corresponding Interaural Time Difference (ITD), it is possible to locate the prey precisely in the horizontal plane, figure inspired by [215], Copyright 2002 Society for Neuroscience). In our system the pMUT emitter (in dark blue) produces a sound wave that bounces off the targeted object. The reflected ultra-sound wave is sensed by 2 pMUT receivers (in light-green) and processed by the neuromorphic processor (right). (b) The ITD computation model (Jeffress model) describes how sounds reaching the barn owl's ears are first encoded into phase-locked spike trains in the Nucleus Magnocellularis (NM), and then processed using a grid of geometrically arranged coincidence detector neurons in the Nucleus Laminaris (NL) (left). Illustration of the neural ITD computational map combining delay lines and coincidence detector neurons that can be implemented using RRAM-based neuromorphic circuits to model the owl's biological sensing system (right). (c) Diagram of the basic Jeffress mechanism, where the two ears receive a sound stimulus at different moments due to a difference in the ToF and send axons to detectors from opposite ends. The axons are afferent to an array of coincident detector neurons (CDs), each of which is selective to highly temporally correlated inputs. As a result, only the CDs whose inputs arrive with the smallest time difference (the ITD is exactly compensated) will be maximally excited. The CDs will then encode the angular position of the target.

Inspired by the auditory system of the barn owl, we developed a biomimetic system for object localization. The two ears are represented by the two pMUT receivers. The sound source is a pMUT emitter located in between (Fig. 3.1a), and the computational map is formed by a grid of RRAM-based CD circuits (Fig. 3.1b, in green) taking the role of CD neurons, whose inputs are delayed by delay line circuits (in blue) which act as the axons in the biological counterpart.

3.2 PMUT sensors for Time-of-Flight measurement.

Piezoelectric micromachined ultrasonic transducers are scalable ultra-sound sensors that can be integrated with advanced CMOS technology[203, 204, 205, 220], and have lower actuating voltage and power consumption than conventional bulk transducers [221]. In our work, the diameter of the membrane is 440μm and the resonance frequency spreads in the range $[110 - 117]$k Hz (Fig. 3.2a). Over a batch of 10 tested devices, the median quality factor is around 50 [203]. Combining the information of different membranes is a well-known technique to infer angular information from pMUT devices, using for instance beamforming techniques [203, 222]. The potential of this technology for air-borne pulse-echo measurements has been demonstrated using a beamforming strategy on a system composed of a pMUT emitter (composed of 1 membrane) and a pMUT receiver system (made of 5 pMUT membranes arranged in an array with a pitch of 1.5 mm), located few centimeters apart from each other[203]. We conducted an experiment locating two pMUT sensors about 10 cm apart from each other, thus fully taking advantage of the different ToF of sound being sensed by the two receiving membranes. A single pMUT working as an emitter is located in between the receivers. A 12 cm-wide PVC plate located in front of the pMUT devices at a distance D was used as a target (Fig. 3.2b). The receivers record the sound reflected from the object and respond maximally at the Time-of-Flight of the sound wave. The experiment was repeated varying the position of the object, defined by its distance D and its angle Θ. Inspired by [223], we propose a neuromorphic pre-processing of the pMUT raw signal, as described in Fig. 3.2c: for each of the two pMUT receivers, the raw signal is band-pass filtered to smooth it out, rectified, and later passed to an Integrate-and-Fire (IF) neuron, which converts the signal into a spike train. This in turn feeds a second Leaky-Integrate-and-Fire (LIF) neuron that produces an output event (spike) in case of overcoming a dynamical threshold (Fig. 3.2d): the timing of the output spike encodes the detected Time-of-Flight. The threshold of the LIF is calibrated to the pMUT response mitigating the pMUT's device-to-device variability. Thanks to this approach, instead of storing the whole sound wave to memory and process it later, we simply generate a spike at the reception of the sound wave, which constitutes the input of the resistive memory-based computational map. To assess the localization angular precision allowed by the pMUTs and the proposed signal processing technique, we measured the ITD (i.e. the time difference between the spike events generated by the two receivers) when varying the distance and angle of the object. The ITD is then analytically converted into an angle and plotted as a function of the object position: the uncertainty

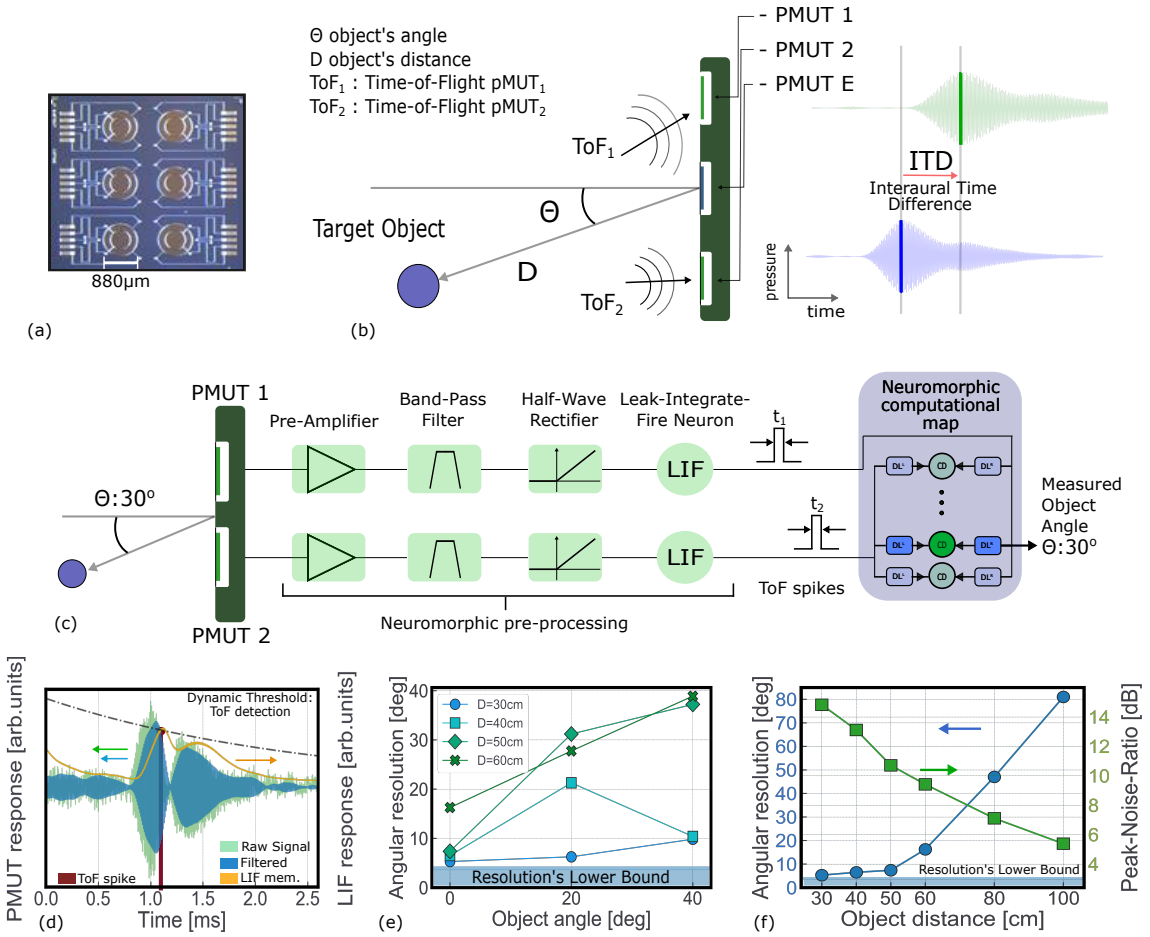


Figure 3.2: Sensory system assessment. (a) Picture of a pMUT die with six 880 μm-diameter membranes integrated with a 1.5 mm pitch. (b) Diagram of the measurement setup. A target object is located at an azimuthal position θ and distance D. An emitter pMUT produces a wave-form at 117.6 kHz that is reflected by the object and arrives at the 2 pMUTs receivers with different Time-of-Flight (ToF). Such difference, defined as Interaural Time Difference (ITD), encodes the position of the object and can be estimated evaluating the peak of the response in the two receiver sensors. (c) Diagram of the pre-processing steps to convert the raw pMUT signal into a train of spikes (*i.e.* the input for the neuromorphic computational map). The pMUT sensors and the neuromorphic computational map have been fabricated and tested, while the neuromorphic pre-processing is based on software simulations. (d) Response of the pMUT membrane upon arrival of a signal and conversion to the spike domain. (e) Experimental angular precision of the localization as a function of the object angle (Θ) and the distance (D) of the target object. The minimum angular resolution imposed by the ITD extraction method is of about 4^o. (f) Angular precision (blue line) and corresponding Peak-to-Noise Ratio (green line) as a function of the object distance for Θ=0.

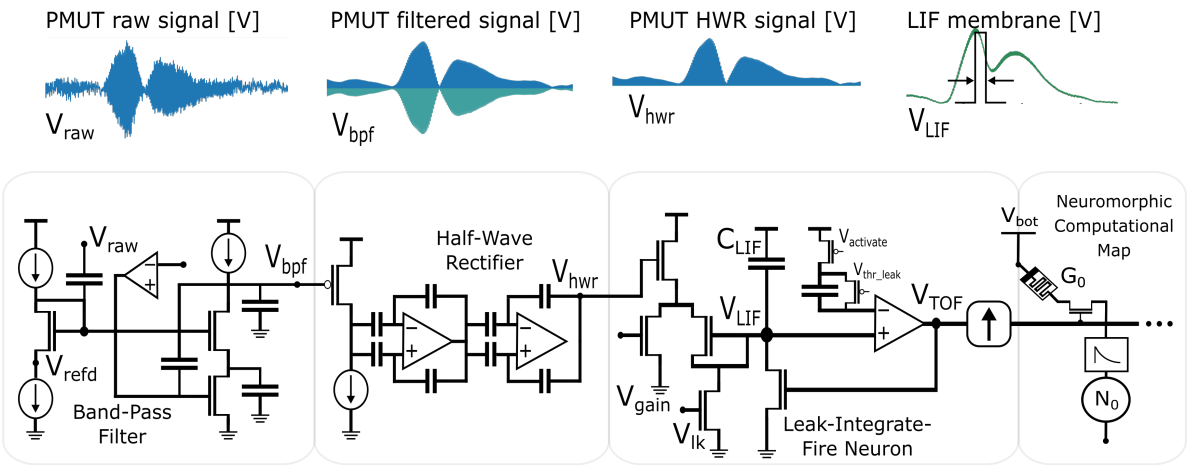


Figure 3.3: PMUT data treatment assessment. From left to right, the Raw signal of the pMUT is converted to a Voltage: at this stage it is hard to retrieve any information from the signal. A Band-Pass Filter (BPF) centered around the pMUT resonance frequency is applied to remove most of the noise and smoothen the signal. Later, the BPF signal is Half-Wave-Rectified to remove the negative component. The resulting signal is feeding a Leaky-Integrate-and-Fire (LIF) neuron. This LIF neuron features an exponentially decreasing threshold which, once overcome, induces a single output spike emitted at the Time-of-Flight of the sound wave detected by the pMUT receiver. Circuits on the lower part of the figure are taken from[223].

over the measured ITDs grows with both the object's distance and angle (Figs. 3.2e and f). The main challenge is the noise in the pMUT response. The more distant the object is located, the higher the noise, thus lowering the Peak-to-Noise ratio (Fig. 3.2f, green line). The decrease in the Peak-to-Noise Ratio (SNR) leads to an increase in the uncertainty over the estimated ITD and consequently on the precision of the localization (Fig. 3.2f, blue line). For an object located 50 cm away from the emitter, the system's angular precision is about 10^o. This limit, imposed by the sensor's characteristics, can be improved. For example, the emitted signal can be strengthened by coupling several emitters, and/or by using multiple receivers to average the information on the detected Time-of-Flight, and thus lowering the uncertainty. This would extend the range of detection, as demonstrated in [206], at the price of an added energy cost.

Acoustic measurement setup and pMUT characterization. pMUT sensors are arranged in a printed circuit board, separating the two receivers of about 10 cm, with the emitter between the receivers. In this work, each membrane is a suspended bimorph structure made of two 800 nm-thick piezoelectric Aluminium Nitride (AlN) layers sandwiched between three 200 nm-thick Molybdenum (Mo) layers, covered by a 200 nm-thick top SiN passivation layer, as reported in [224]. Inner and outer electrodes are patterned

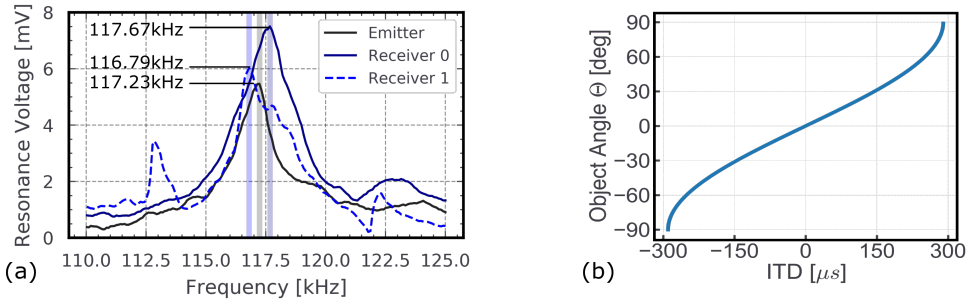


Figure 3.4: PMUT characterization and Object Angle as a function of ITD. (a) Resonance voltage amplitude of the 3 pMUT membranes as a function of the input driving voltage frequency. Each membrane shows a slightly different resonance frequency, due to imperfections in fabrication. This mismatch does not affect the quality of the measurements. (b) Analytical relationship between the Interural Time Difference (ITD) and angular/azimuthal object position.

on the bottom and the top Mo layers, while the middle Mo electrode is not patterned and used as the ground, resulting in a membrane with four electrodes pairs. This architecture enables to exploit the whole deformation of the membrane, resulting in a enhanced drive and receive sensitivity. Such a pMUT typically presents a drive sensitivity of typically 700 nm/V as an emitter, delivering a surface pressure of 270 Pa/V. As a receiver, a single pMUT membrane presents a short-circuit sensitivity of 15 nA/Pa, directly related to the piezoelectric coefficients of AlN. The technological variability of the stress within the AlN layers results in a resonant frequency variation which is compensated by applying a DC bias to the pMUT. The DC sensitivity has been measured at 0.5 kHz/V. For acoustic characterization, a microphone is used in front of the pMUT. For pulse-echo measurements, we positioned a rectangular plate of about 50 cm^2 in front of the pMUTs, reflecting the emitted sound wave. Both the distance of the plate and the angle with respect to the pMUT plane are controlled utilizing dedicated supports. Tectronix CPX400DP voltage sources bias the three pMUT membranes to tune the resonant frequency to 117.6 kHz[203], while the emitter is controlled by a Tectronix AFG 3102 pulse generator set close to the resonance frequency (117.6 kHz), and a duty cycle of 0.01. The currents read at the 4 output ports of each pMUT receiver is converted into a voltage by a dedicated differential current-to-voltage architecture and the resulting signal is digitized by a Spektrum acquisition system. We characterized the limit of detection by collecting the pMUT signal in different conditions: we moved the reflecting plate at different distances [30,40,50,60,80,100] cm and varied the angle of the pMUT support ([0, 20,40] ^{o}). Fig. 3.4b shows the relationship between the temporal resolution in detecting Interaural Time Difference (ITD) and the corresponding angular position in degrees.

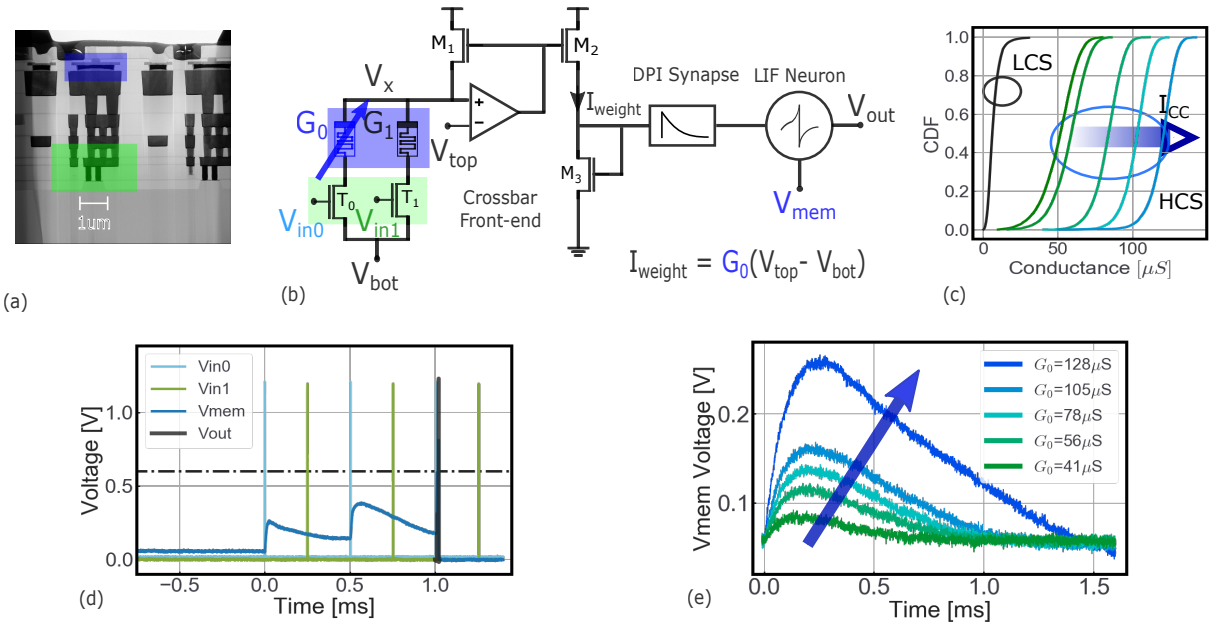


Figure 3.5: Role of RRAM devices in neuromorphic circuits. (a) Scanning Electron Microscopy (SEM) image of a HfO_2 1T1R RRAM device, in blue, integrated on 130 nm CMOS technology, with its selector transistor (width of 650 nm) in green. (b) Basic building block of the proposed neuromorphic circuit. Inputs voltage pulses (spikes), V_{in0} and V_{in1}, draw a current I_{weight} proportional to the conductance states, G_0 and G_1, of the 1T1R structures. This current is injected into a DPI synapse that excites a LIF neuron. The RRAMs G_0 and G_1 are set in the HCS and LCS respectively. (c) Cumulative Density Function of the conductance of a population of 16 kb RRAM devices, as a function of the compliance current I_{CC}, which effectively controls the conductance level. (d) Measurement of the circuit in (a), showing that G_1 (in LCS) effectively blocks inputs from V_{in1} (green), in fact the output neuron's membrane voltage only responds to the blue input of V_{in0}. RRAMs efficiently define the connections in the circuit. (e) Measurement of circuit in (b) showing the effect of the conductance value G_0 on the membrane voltage V_{mem}, following the application of a voltage pulse Vin_0. The larger the conductance, the stronger the response: the RRAM device thus implements the weight of the input-to-output connection. Measurements have been performed on one circuit and demonstrate the dual function of the RRAMs, they route and weigh the input pulses.

3.3 RRAM-based neuromorphic computational map.

Resistive memories store information in their non-volatile conductive states. The basic working principle of this technology is that modifying a material at the atomic level results in changes of its conductance [225]. Here we use an oxide-based resistive memory composed of a 5 nm hafnium-dioxide layer sandwiched between a top and a bottom electrode of titanium and titanium nitride. The conductivity of an RRAM device can

be modified by the application of current/voltage waveforms, which create or break a conductive filament composed of Oxygen vacancies between the electrodes. We have co-integrated such devices in a standard 130 nm CMOS process [226] to build a fabricated re-configurable neuromorphic circuit implementing coincidence detectors and the delay lines circuits (Fig. 3.5a). Both the non-volatility and analog nature of the devices perfectly couple with the event-driven nature of the neuromorphic circuits, minimizing power consumption when idle. The basic building block of the proposed circuit is presented in Fig. 3.5b. It is composed of N parallel one-resistor-one-transistor (1T1R) structures, encoding the synaptic weights, from which a weighted current is extracted and then injected to a common differential pair integrator (DPI) synapse [70], and finally into a Leaky-Integrate-and-Fire (LIF) neuron [23].

RRAM-based neuromorphic circuits present the challenge of designing a fully analog electronic system in which the RRAM devices coexist with conventional CMOS technology. In particular, the conductive state of the RRAM devices has to be read and utilized as a functional variable of the system. To do so, a circuit that reads a current from a device upon arrival of an input pulse and that uses such current to weight the response of a Differential Pair Integrator (DPI) synapse has been designed, fabricated, and tested. The circuit is shown in Fig. 3.5a and it represents the basic building block of the neuromorphic platform in Fig. 3.6a. The input spikes are applied at the gates of the 1T1R structures as trains of voltage pulses, with a pulse-width on the order of hundreds of nanoseconds. Input pulses result in a current flow through the RRAM proportional to the conductance of the device, G ($I_{weight} = G(V_{top} - V_x)$). The operational amplifier circuit (OPAMP) has a constant DC bias voltage V_{Top} applied to its inverting input. The negative feedback of the OPAMP will act to ensure that $V_x = V_{top}$, by sourcing an equal current from transistor M_1. The current extracted from the device, I_{weight}, is injected onto the DPI synapse. Stronger currents will result in greater depolarization, thus the RRAM's conductance effectively implements the synaptic weight. This exponential synaptic current is injected onto the membrane capacitor of a leaky-integrate and fire (LIF) neuron where it integrates as a voltage. If the threshold voltage of the membrane (the switching voltage of an inverter) is overcome, the output section of the neuron is activated, producing an output spike. This pulse feeds back and shunts the neuron membrane capacitor to ground such that it is discharged. The circuit is then complemented by a pulse extender, not shown in Fig. 3.5a, that reshapes the output pulse of the LIF neuron to the target pulse width. Further multiplexers were integrated on each line in order to be able to apply voltages to the top and bottom electrodes of the RRAM devices.

The resistive memories can be SET into a high conductance state (HCS) by applying an external positive voltage reference on V_{top} while grounding V_{bottom}, and RESET into a low conductive state (LCS) by applying a positive voltage on V_{bottom} while grounding V_{top}. The mean value of the HCS can be controlled by limiting the SET programming (compliance) current (I_{CC}) via the gate-source voltage of the series transistor (Fig. 3.5c). The function of RRAMs in the circuit is dual: they route and weigh input pulses.

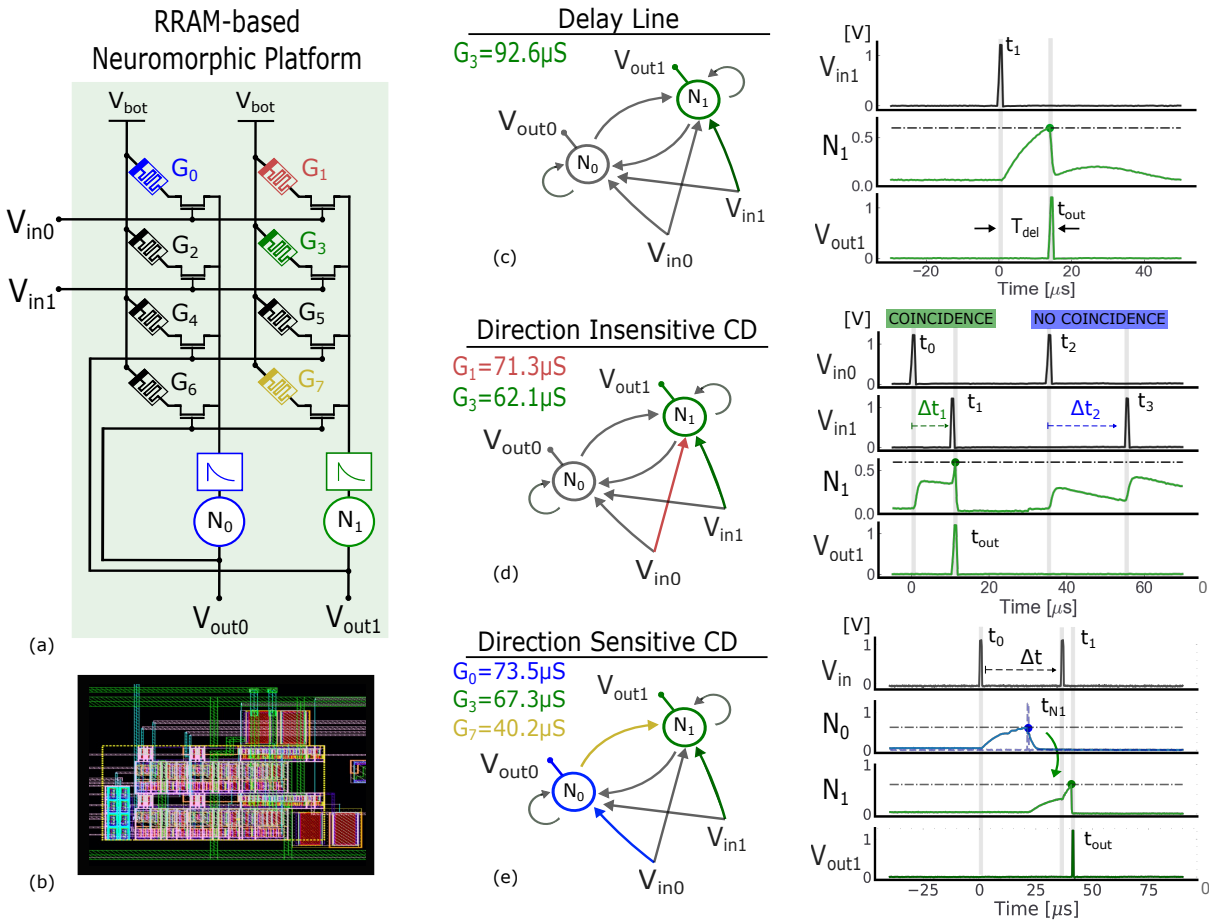


Figure 3.6: Experimental measurements of the RRAM-based neuromorphic circuital platform. (a) Diagram of the circuit formed by two output neurons, N_0 and N_1, receiving two inputs 0 and 1. The four devices on the top of the array define the synaptic connections from inputs to outputs and the four cells on the bottom define the recurrent connections among neurons. Colored RRAMs represent devices set in the HCS on the diagrams to the right: a device in the HCS allows a connection to be formed and expresses a weight, while a device in the LCS blocks the input pulse and disables the connection to the output. (b) Layout of the circuit in (a), with the 8 RRAMs highlighted in blue. (c) A delay line is formed by simply exploiting the dynamics of a DPI Synapse and a LIF Neuron. The green RRAM is set to high enough conductance to allow the output spike to be elicited following the input spike by a delay Δt. (d) Diagram of the direction insensitive CD detecting temporally correlated signals. The output neuron 1, N_1, spikes upon arrival of input 0 and input 1 with small delay. (e) Diagram of the direction-sensitive CD, a circuit that detects when input 1 arrives in close proximity and after input 0. The output of the circuit is represented by neuron 1 (N_1).

First, thanks to the two main conductive states (HCS and LCS), the RRAMs can

either block or pass the input pulses when they are respectively in the LCS or HCS state. As a consequence, RRAMs efficiently define the connections in the circuit. This is fundamental to allow the architecture to be re-configurable. In order to prove that, we characterized a fabricated circuit implementation of the circuit block in Fig. 3.5b. An RRAM corresponding to G_0 was programmed into the HCS and a second RRAM, G_1, was programmed in the LCS. Input pulses were applied to both V_{in0} and V_{in1}. The effect of two input trains of pulses was analyzed in the output neuron, by collecting the membrane voltage and output of the neuron with an oscilloscope. The experiment is successful when only the pulses connected to the neuron by the HCS device (G_0) excite the membrane voltage. This is demonstrated in Fig. 3.5d, where the blue train of pulses makes the membrane voltage accumulate charge on the membrane capacitor, whereas the green train of spikes leaves the membrane voltage unperturbed.

The second important function of RRAMs is implementing the weight of the connections. By exploiting the analog adjustment of the RRAMs conductance, the input-to-output connection can be appropriately weighted. In a second experiment, the device G_0 is programmed in different HCS levels and an input pulse is applied to the input V_{In0}. The input pulse extracts a current from the device (I_{weight}) which is proportional to the conductance and the corresponding potential drop $V_{top} - V_{bot}$. This weighted current is then injected into the DPI synapse and output LIF neuron. The membrane voltage of the output neuron is recorded with an oscilloscope and plotted in Fig. 3.5e. The peak of the neuron membrane voltage responding to a single input pulse is proportional to the conductance of the resistive memory, confirming that RRAMs can be exploited as programmable synaptic weight elements. These two preliminary tests demonstrate that the proposed RRAM-based neuromorphic platform is able to implement the basic elements of the Jeffress basic mechanism, namely the delay line and coincidence detector circuits. The circuital platform is constituted by stacking consecutive blocks, as the one in Fig. 3.5b, side by side and connecting their gates to common input lines. We designed, fabricated, and tested a neuromorphic platform composed of two output neurons and receiving two inputs (Fig. 3.6a). The layout of the circuit is shown in Fig. 3.6b. The upper 2×2 RRAM matrix allows to route the input pulses to the two output neurons, while the lower 2×2 array allows the two neurons (N_0, N_1) to be recurrently connected. We demonstrate that this platform can assume a delay line configuration and two distinct coincidence detector functionalities, as summarized by the SPICE simulations in Fig. 3.6c, d, and e.

The Delay Line (Fig. 3.6c) simply exploits the dynamical behavior of the DPI synapse and LIF neuron to reproduce the input spike from V_{in1} to V_{out1} with a delay T_{del}. Only the RRAM connecting V_{in1} to V_{out1}, G_3, is programmed into the HCS, while the other RRAMs are in the LCS. The G_3 device is programmed to 140 μS to ensure that each input spike increases the membrane voltage of the output neuron sufficiently to reach the threshold and to generate a delayed output spike. The delay T_{del} is defined by both the synapse and neuron time constants. A coincidence detector detects the occurrence of temporally correlated but spatially distributed input signals. A direction insensitive

CD relies on separate inputs converging to a common output neuron (Fig. 3.6d). The two RRAMs connecting V_{in0} and V_{in1} to V_{out1}, G_2 and G_4 respectively, are programmed into the high conductance state. Synchronous arrival of spikes at V_{in0} and V_{in1} pushes the membrane voltage of the neuron N_1 over the threshold required to generate an output spike. If the two inputs arrive too far apart in time, the charge on the membrane voltage accumulated by the first input may have time to decay away, preventing the membrane potential of N_1 to reach the threshold. The G_1 and G_2 are programmed to 70 μS, ensuring that a single input spike does not increase the membrane voltage enough to generate an output spike. The direction-sensitive CD is a circuit sensitive to the spatial order of arrival of impulses: from right to left or vice versa. This is a basic building block in the elementary motion detection network of Drosophila's visual system to compute the direction of motion and detection of collisions [227]. To implement a direction-sensitive CD the two inputs have to be routed to two different Neurons (N_0, N_1) and between those, a directional connection has to be established (Fig. 3.6e). Upon the arrival of the first input, N_0 responds by increasing its membrane voltage up to overcoming its threshold and emitting a spike. Thanks to the directional connection in green, this output event in turn excites N_1. If the V_{in1} input event arrives to excite N_1 when its membrane voltage is still high, N_1 will produce an output event, signifying the detection of coincidence between the two inputs. The directional connection allows N_1 to emit an output only if input 1 arrives after input 0. The G_0, G_3, and G_7 are respectively programmed to 140 μS, 70 μS, and 70 μS, ensuring that a single input spike at V_{in0} generates a delayed output spike, while the membrane potential of N_1 reaches the threshold only upon the synchronous arrival of two input spikes.

Circuit measurement setup and RRAM characterization. The electrical tests involved analysing and recording the dynamical behavior of analog circuits as well as programming and reading RRAM devices. Both phases required dedicated instrumentation, all simultaneously connected to the probe card. RRAMs devices in the neuromorphic circuits are accessed from the external instrumentation by means of multiplexars (MUXs). The MUXs decouple the 1T1R cell from the rest of the circuit where they belong, allowing to read and/or program the device. For programming and reading the RRAM devices, a Keithley 4200 SCS machine was used combined with an Arduino microcontroller: the first for precise pulse generation and current reading, the second to fast access a single 1T1R element in the memory array. The first operation is the forming of the RRAM devices. The cells are selected one by one and a positive voltage was applied between the top and bottom electrodes. At the same time, the current is limited to the order of tens of micro-amperes by applying an appropriate gate voltage to the selector transistor. Afterwards, the RRAM cells can be cycled between the Low Conductance State (LCS) and the High Conductance State (HCS) through RESET and SET operations, respectively. SET operations are performed with a positive square voltage pulse of $1\mu s$ width and $[2.0 - 2.5]V$ peak voltage applied to the Top Electrode, and a similarly shaped synchronous pulse with $[0.9 - 1.3]V$ peak voltage applied to the gate of the selector transistor. Such

values allow to modulate the RRAM conductance in the $[20 - 150]\mu S$ interval. For the RESET, a pulse of $1\mu s$ width and 3V peak is applied to the Bottom Electrode (Bit Line) of the cell while the gate voltage is in the $[2.5 - 3.0]$V range. Inputs and outputs of the analog circuits are dynamical signals. In the case of the input, we have alternated two HP 8110 pulse generators with a Tektronix AFG3011 waveform generator. Input pulses have a width of $1\mu s$ and rise/fall edge of 50 ns. This type of pulse is assumed as the stereotypical spiking event in the spike-based analog circuit. Concerning the outputs, a 1 GHz Teledyne LeCroy oscilloscope was utilized to record the output signals. The acquisition speed of the oscilloscope has been proven not to be a limiting factor analysing and collecting data from the circuits.

3.3.1 Variability in neuromorphic circuits and RRAM-based calibration procedure.

Variability is a common source of non-ideality in analog neuromorphic systems [142, 228, 229]. It results in heterogeneous behaviors among neurons and synapses. Examples of such imperfections include for example 30% (mean value over standard deviation) of variability on input gain, time constants and refractory period, to name a few. This issue is more pronounced when several neuron circuits are connected together, as in the case of the direction-sensitive CD, which consists of two neurons. To function properly, the input gain and decay time constants of the two neurons should be as similar as possible. For example, large differences in input gain may result in a neuron responding excessively to an input pulse, while the other being almost insensitive. Fig. 3.8a shows that randomly selected neurons respond differently to the same input pulse. This neuron variability has an impact on e.g. the functionality of the direction-sensitive CD. In the circuit characterized in Fig. 3.8b and c, neuron 1 presents a much higher input gain than neuron 0. As a result, neuron 0 requires 3 input pulses (instead of 1) to reach the threshold, while neuron 1 reaches the threshold with two input events, as expected. We propose to exploit the plastic behavior of the resistive memory as a mean of acting on the input gain of neurons and reduce the impact of the neuromorphic circuit variability.

3.3.2 Mitigating RRAM variability with a dedicated programming strategy

Resistive Random Access Memories are electronic devices based on the formation and rupture of a conductive filament across an insulator material. Here we consider an oxide-based resistive memory composed of 5 nm of hafnium-dioxide sandwiched between a top and bottom electrode of titanium and titanium-nitride. The cells can be programmed in different conductance states by applying appropriate voltage and current waveforms over the device. The change in the geometry of the filament results in different conductance states in the device.

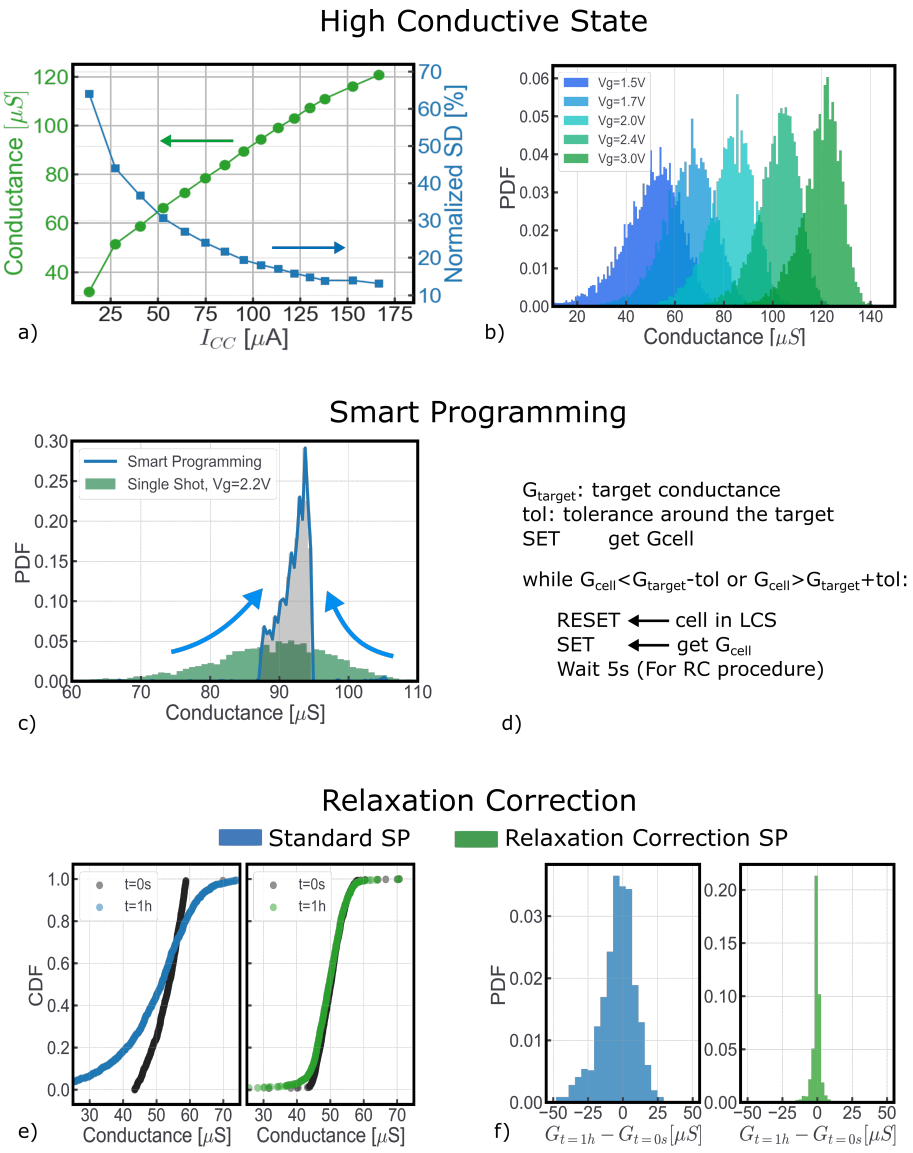


Figure 3.7: Characterization of the RRAM devices from a 16kb array. a) HCS conductance and Normalized Standard Deviation as a function of the compliance current. b) Visualization of some distribution of HCS states with different Compliance current. c) Smart Programming operation solving the problem of variability of the conductive state. d) Description of the recursive programming operation called Smart Programming, with a hint of the Relaxation Correction procedure. e, f) Effect of the Relaxation Correction on the SP procedure. Adding a 5s waiting time between programming iterations reduced the effect of Relaxation and Retention. Left, difference between Standard SP (blue) and RCSP (green) on 1096 devices measured after 1h. Right, distribution of the conductance difference between those measured after programming ($G_{t=0s}$) and after 1 hour ($G_{t=1h}$).

By applying a positive/negative SET/REST voltage between top and bottom electrode, a conductive filament is formed/disrupted, thus increasing/decreasing the conductance. When the filament is formed the cell is in the High Conductive State (HCS), otherwise the cell is in the Low Conductive State (LCS). Resistive memory technologies suffer from conductance variability: if a device is repeatedly cycled under the same programming conditions a conductance distribution emerges. The median conductance and the standard deviation of this distribution are determined by the compliance current (i.e. programming current) applied during the SET operation (I_{CC}). Figs. 3.7a and b show the modulation of the median conductance and standard deviation of the HCS distributions as a function of the compliance current. The measurements have been performed on a 16 kb array of one transistor-one resistor (1T1R) devices. The cells are SET using 15 programming conditions (V_{GS} in $[1.4 - 3.0]V$), resulting in different compliance currents, and $2V$ on the top electrode. The cycle to cycle variability prevents programming of an RRAM cell to a precise conductance value. Numerous techniques have been developed to cope with cycle-to-cycle variability when programming RRAM devices. In [129], a technique has been developed to yield up to 8 different separated levels of conductance from a single RRAM device, by means of an iterative procedure alternating Resets and Set operations. We will refer to it as Smart Programming (SP). Fig. 3.7c shows the difference between a single shot SET operation and the Smart Programming applied to the 16kb array, the latter obtaining a much tighter distribution of devices. The Smart Programming procedure involves recursive programming of a single device, alternating carefully modulated Set and Reset operations, until the conductance of the device falls within a predetermined target range. Nonetheless, this procedure still suffers from relaxation: the conductance distribution obtained just after programming (t=0, black in Fig. 3.7c) rapidly spreads toward both higher and lower values. After 1 hour more than 45% of the programmed devices are out of the target conductance range. A fix for this problem is found in the Relaxation Correction Smart Programming procedure: the flow of this operation is the same as in the standard SP, but a waiting time of 5 s is added between each re-programming operation. Cells that suffer from conductance instability during the waiting period are rescheduled to the next programming iteration, allowing the algorithm to take into account and correct the conductance relaxation. Fig. 3.7d shows the difference between Standard SP (blue) and Relaxation Correction SP (green) on 1096 devices programmed with the same target conductance interval. The black line represents the tight conductance distribution right after the last programming operation (t=0), and the colored ones are measured after 1 hour. Relaxation Correction SP greatly reduces the temporal variability. Fig. 3.7e visualizes the same data plotting the difference of the conductance after 1h with that at $t = 0s$ ($G_{1h} - G_{0s}$). This highlights how single devices were changing their conductance over the course of 1 hour. The green distribution is much tighter that the blue one, meaning that the Relaxation Correction procedure effectively reduced the temporal instability.

3.3.3 A calibration procedure to minimize the impact of variability at the system level

The analog programming property of RRAMs can be exploited to mitigate the problem of variability in analog neuromorphic circuits. We developed a simple calibration procedure that consists in re-programming the RRAM device until the circuit under analysis meets certain requirements. For a given input, the output is monitored and the RRAMs are re-programmed until the target behavior is achieved. A 5s waiting time is introduced between programming operations to mitigate the RRAM Relaxation issue, causing temporal fluctuations of the conductance. The synaptic weights are adapted or calibrated to the requirements of the analog neuromorphic circuit. Focusing on the two basic functionalities of the Neuromorphic Platform, the delay lines and direction insensitive CD, the calibration procedure is summarized in Algorithms [1, 2]. For the delay line circuit, the target behavior is to deliver the output pulse with a delay Δt. If the actual delay of the circuit is smaller than the target, the synaptic weight G_3 has to be decreased (the G_3 has to be RESET and then SET with a lower compliance current, I_{cc}). Contrarily, if the actual delay is longer than the target, the conductance of G_3 has to be reinforced (the G_3 has to be RESET and then SET with a higher I_{cc}). This procedure is repeated until the delay produced by the circuit matches the target, with a tolerance set to stop the calibration procedure. For the direction insensitive CD, the calibration procedure involves two RRAM devices, G_1 and G_3. The circuit is provided with two inputs, V_{in0} and V_{in1} delayed by dt. The circuit must only respond to delays lower than the coincidence range [0,dt_{CD}]. When an output spike is absent whereas the input spikes are close, both the RRAM devices must be reinforced in order to help the neuron reach the threshold. Conversely, if the circuit responds to delays larger than the target range dt_{CD}, the conductances have to be decreased. The procedure is repeated until the correct behavior is obtained. The compliance current can be modulated by the embedded analog circuit presented in [230, 231]. Exploiting this embedded circuit, one could perform such procedure periodically to calibrate the system or to re-purpose it for different applications.

Algorithm 1: Delay Line RRAM calibration

$G_0 \leftarrow SET_{HCS}(I_{CC})$
T_{del} : target delay
$t_{del} \leftarrow$ apply input pulse
tol: tolerance for the delay

while $t_{del} < T_{del} - tol$ or $t_{del} > T_{del} + tol$:
    $t_{del} \leftarrow$ apply input pulse
    if $t_{del} > T_{del} + tol$:
        $I_{CC}^{'} = I_{CC} + \Delta I$
        $G_0 \leftarrow SET_{HCS}(I_{CC}^{'})$
        Wait 5s (Relaxation correction)
    elif $t_{del} < T_{del} - tol$:
        $I_{CC}^{'} = I_{CC} - \Delta I$
        $G_0 \leftarrow SET_{HCS}(I_{CC}^{'})$
    end
end

Algorithm 2: Direction Insensitive CD calibration

$G_0, G_1 \leftarrow SET_{HCS}(I_{CC}^{'})$
$[0, \Delta t_{CD}]$: target detection range

$CD_{response} \leftarrow$ apply input pulses

while CD does not respond to $[0, \Delta t_C D]$ only:
    $CD_{response} \leftarrow$ apply input pulses

    if CD not responding to dt in $[0, \Delta t_{CD}]$:
        $I_{CC}^{'} = I_{CC} + \Delta I$
        $G_0, G_1 \leftarrow SET_{HCS}(I_{CC}^{'})$
        Wait 5s (Relaxation correction)

    elif CD responding to dt in $dt > \Delta t_{CD}$:
        $I_{CC}^{'} = I_{CC} - \Delta I$
        $G_0, G_1 \leftarrow SET_{HCS}(I_{CC}^{'})$
    end
end

The two elements employed in the ITD computational map are the delay lines and direction insensitive CD. Both circuits need to be precisely calibrated to ensure good performance of the object localization system. The delay line has to precisely deliver a delayed-version of the input spike (Fig. 3.9a), the CD must be activated only when the inputs fall within the target detection range. For the delay line, the synaptic weight of the input connection (G_3 in Fig. 3.6a) is re-programmed until the target delay is obtained. A tolerance around the target delay is set to stop the procedure: the smaller the tolerance, the harder it is to successfully tune the delay line. Fig. 3.9b shows the result of the calibration procedure for the delay line: as it can be seen the proposed circuit can provide all the delays required in the computational map (from 10 to 300 μs). The maximum number of calibration iterations affects the quality of the calibration procedure: 200 iterations allow the error to be reduced to less than 5%. One calibration iteration corresponds to a SET/RESET operation of the RRAM cell. The tuning procedure is crucial also to improve the accuracy of the detection of the temporally close events of the CD module. Ten calibration iterations are needed to reach a true positive rate (*i.e.* rate of events correctly detected as correlated) higher than 95% (blue line in Fig. 3.9c). However, the tuning procedure has no effect on false positive events (*i.e.* rate of events incorrectly detected as correlated). Another technique observed in biological systems which solves the time constraint of rapidly activated pathways is redundancy (*i.e.* many copies of the same entity are used to fulfill a given function). Taking inspiration from biology [232], we stacked multiple CD circuits in each CD module between two delay lines to reduce the effect of False Positive detection. As shown in Fig. 3.9c (green lines) stacking 3 CD

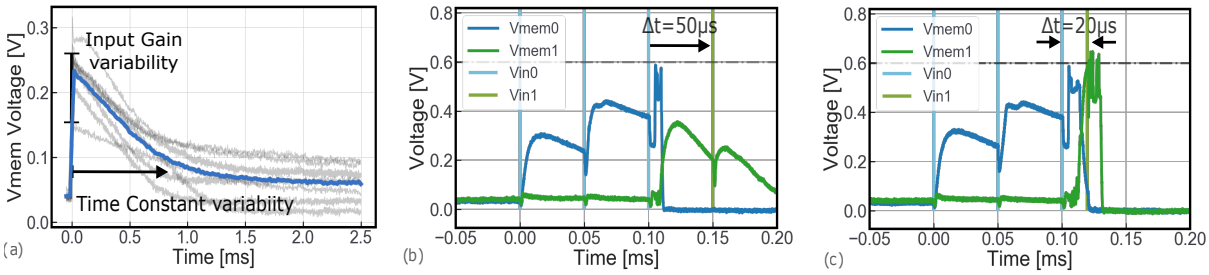


Figure 3.8: Variability in analog neuromorphic circuits. (a) Experimental measurements of the response of 9 randomly selected individual neurons to the same input spike. The response varies across the population impacting both the input gain and the time constant. (b) Experimental measurements of the impact of the neuron to neuron variability affecting the direction-sensitive CD. Due to the neuron to neuron variability the two output neurons of the direction-sensitive CD respond differently to the input stimuli. Neuron 0 presents a lower input gain than neuron 1, thus requiring 3 input pulses (instead of 1) to produce an output spike. Neuron 1 reaches the threshold with two input events, as expected. If input 1 arrives $\Delta t = 50\mu s$ after neuron 0 has been excited, the CD remains silent because Δt is larger than neuron 1's time constant (about $22\mu s$). (c) Decreasing the $\Delta t = 20\mu s$, makes input 1's spike to arrive when neuron 1's excitation is still high, resulting in the coincidence detection of the two input events.

elements in each CD module allows reducing the False Positive rate to less than 10^{-2}.

3.4 System assessment

We now assess the performance and energy consumption of the object localization system using the results of the electrical characterization of the pMUT sensors, the CD, and the delay line circuits composing the neuromorphic computational map inspired by the Jeffress model (Fig. 3.1 a). Regarding the neuromorphic computational map, the higher the number of CD modules, the better the angular resolution, but also the higher the system energy (Fig. 3.10a). A trade-off is reached by comparing the precision of the single components (both pMUT sensors, and neuron and synapse circuits) with that of the whole system. The resolution of delay lines are limited by the time constants of the analog synapses and neurons, which are greater than 10 μs in our circuits, corresponding to an angular resolution of 4^o. A more advanced CMOS technology node would enable the design of neuron and synapse circuits with lower time constants and consequently higher precision of the delay line element. However, in our system, the precision is limited by the pMUT uncertainty in the estimation of the angular position, that is 10^o (dark-blue horizontal line in Fig. 3.10a). We fixed the number of CD modules to 40, corresponding to an angular resolution of about 4^o, that is the computational map angular precision (light-blue horizontal line in Fig. 3.10a). At the system level, this results in a 4^o resolution and 10^o

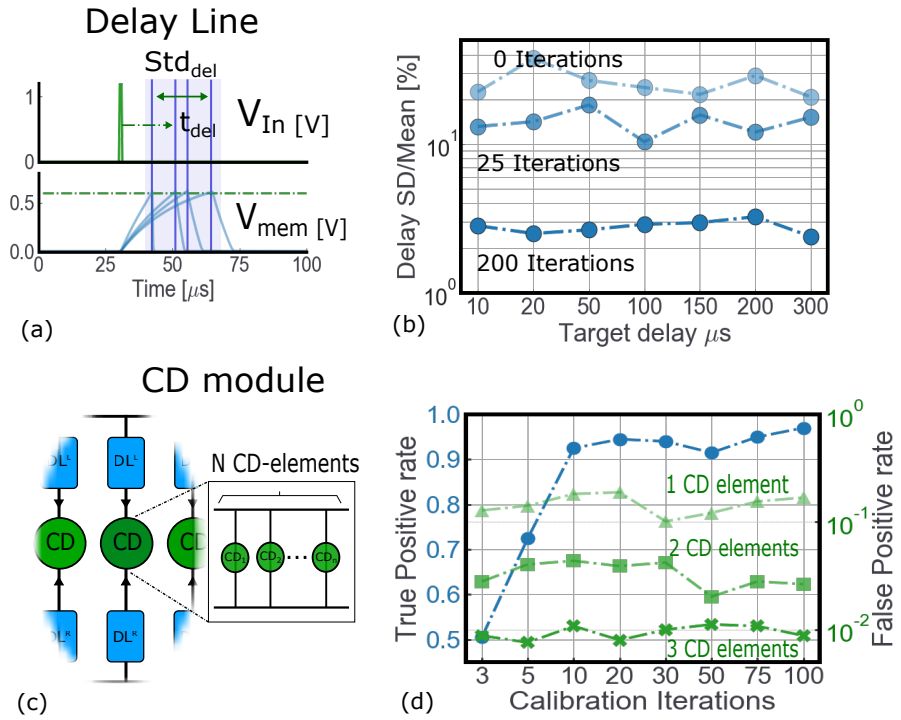


Figure 3.9: Performance of the delay line and direction insensitive CD circuits and impact of the RRAM calibration procedure. (a) Impact of the neuron variability on the delay line circuit. (b) The delay line circuit can be scaled up to larger delays, setting the time constant of the corresponding LIF neuron and DPI synapse to larger values. Increasing the iterations of the RRAM calibration procedure enabled us to substantially improve the precision on the target delay: 200 iterations allow the error to be reduce to less than 5%. One iteration corresponds to a SET/RESET operation on the RRAM cell. (c) Each CD module in the Jeffress Model can be implemented with N-parallel CD elements, to be more resilient to system failures. (d) More RRAM calibration iterations allow to improve the True Positive rate (blue line), while the False Positive rate is independent of the number of iterations (green lines). Stacking more CD elements in parallel enabled us to avoid False Coincidence Detection from a CD Module.

precision for an object located in front of the sensory system at a distance of 50 cm. The single bank power consumption for the pre-processing of the pMUT signal is evaluated at 12.3 nW, according to [223]. Accounting for the 40 CD modules in the computational map, the energy per operation (*i.e.* energy to localize an object) estimated by SPICE simulations is 21.6 nJ. The neuromorphic system is activated only at the arrival of an input event, *i.e.* when the sound wave reaches any of of the pMUT receivers and overcomes the detection threshold, and kept idle otherwise. This allows avoiding unnecessary energy consumption when no input signal is present. Considering a rate of localization operations of 100 Hz and an activation period of 300 μs per operation (maximum possible ITD), the power

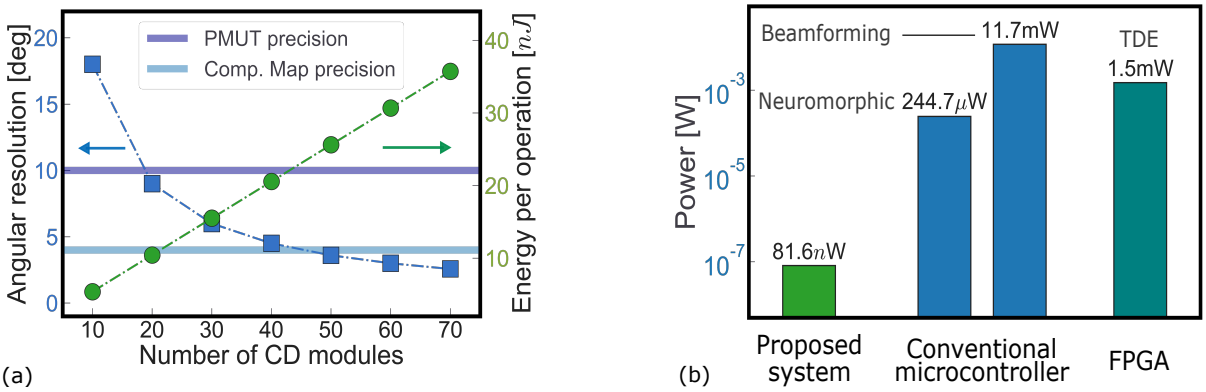


Figure 3.10: Power consumption and angular resolution of the presented neuromorphic sensory and signal processing system. (a) Angular resolution (blue) and energy consumption (green) of one localization operation as a function of the number of CD modules. The dark-blue horizontal bar represents the angular precision of the PMUTs while the light-blue horizontal bar represents the angular precision of a the neuromorphic computational map. (b) Power consumption of the proposed system and comparison with the two discussed implementations on a microcontroller and the digital implementation on an FPGA of the Temporal-Difference-Encoder (TDE)[234]

.

consumption of the neuromorphic computational map results being of 61.7 nW. Accounting for the neuromorphic pre-processing applied to each of the pMUT receivers brings the total system's power consumption at 86.3 nW. To gain a perspective on the energy efficiency of the proposed neuromorphic approach compared to conventional hardware, we benchmark this figure to the energy required for running the same task on a state-of-the-art low-power microcontroller [233] using either neuromorphic or conventional beamforming techniques. The neuromorphic method accounts for an Analog-Digital-Converter (ADC) stage followed by a Band-Pass filter and an envelope extraction stage (Teager-Kaiser method). Finally, a thresholding operation is performed to extract the ToF. We omit the computation of the ITD based on the ToF and the conversion to the estimated angular position as it happens once per measurement. Assuming a sampling frequency of 250 kHz on the two channels (pMUT receivers), 18 operations for the band-pass filter, 3 operations for the envelope extraction and 1 operation for the thresholding per sample, the estimate of the overall power consumption leads to 0.9 mW. The power consumption for the beamforming signal processing solution proposed in [203], accounting for 5 pMUT receivers and 11 beams equally distributed in the $[-50^o, +50^o]$ azimuthal plane is 11.71 mW.

Estimation of the power consumption on a microcontroller We estimate the power consumption of a neuromorphic signal processing approach on an off-the-shelf 32-bits microcontroller [233]. In this estimation, we assumed to operate with an identical setup to the one presented in this work, with 1 pMUT emitter and 2 pMUT receivers.

The method accounts for a Band-Pass filter followed by an envelope extraction stage (Teager-Kaiser method), and finally a thresholding operation applied to the signal to extract the Time-of-Flight. The computation of the ITD and its conversion to the detected angle are omitted in the estimation. We consider the Band-Pass filter to be implemented with a 4^{th} order Infinite Impulse Response filter, requiring 18 floating-point operations. Envelope extraction makes use of further 3 floating-point operations, and a final operation is due for thresholding. In total, 22 floating-point operations are required for pre-processing the signal. The signal sent is a short pulse of a 117.6 kHz sine wave, produced every 10 ms, resulting in a 100 Hz localization operation frequency. We adopt a 250 kHz sample rate to respect the Nyquist theorem and a 6 ms window per measurement to capture a range of 1 m. Note that 6 ms is the Time-of-Flight for an object located at 1 m distance. This gives a power consumption of 180 µW for the analog-to-digital conversion of 0.5 MSPS. The pre-processing of the signal accounts for 6.60 MIPS (instructions per second), resulting in 0.811 mW. However, the micro-processor can be switched to the Low-Power-Mode[235] while not performing the algorithm's operations. This mode allows for a static 10.8 μW power consumption and has a wake-up time of 113μs. Considering the clock rate of 84 MHz, the microprocessor terminates all the operations of neuromorphic algorithm well within the 10 ms period, with a 6.3% duty cycle for the algorithm computation, thus taking advantage of the Low-Power-Mode. The resulting power consumption is of 244.7μW (233.9 μW dynamic and 10.8 μW static power consumption). Adding those two contributions makes for a total of 0.93 mW of power consumption. Note that we omit the derivation of the ITD from the ToFs and the conversion to the detected angle, thus underestimating the power consumption in the microcontroller. This gives further value to the energy efficiency of the proposed system. As a further term of comparison, we estimate the power consumption of a classical beamforming approach, presented in [203, 222], when embedded on the same microcontroller [233] under a 1.8 V supply voltage. Five evenly spaced pMUT membranes are used to provide data for the beamforming. About the processing itself, the beamforming technique used is the Delay-and-Sum. It simply consists in applying a delay to the channels corresponding to the expected time difference of arrival between one channel and a reference channel. If the signals are in phase, once time-shifted, the sum of these signals will exhibit high energy. If they are not in phase, destructive interference will limit the energy of their sum. In [203], a 2 MHz sample rate is chosen to time-shift the data by an integer number of samples. A more frugal approach consists in keeping a coarser 250 kHz sample rate and using Finite-Impulse-Response (FIR) filters to synthesize fractional delays. We will consider that the beamforming algorithm complexity is dominated by the time-shifting because of the convolution of each channel with a 16 taps FIR filter for each direction. To calculate the number of MIPS required for this operation, we consider using a 6 ms window per measurement to capture a 1-meter range, 5 channels, 11 beamforming directions (+/-50° range with a 10° step). Already 75 measurements per second push the microcontroller to its maximum of 100 MIPS. According to [233], this results in a power consumption of 11.26 mW, which gives a total power consumption of 11.71 mW when adding the on-chip ADC contribution.

Estimation of the power consumption on the RRAM-based analog system Based on these estimations, the proposed neuromorphic approach achieves a reduction of five orders of magnitudes in power consumption relative to a microcontroller adopting a classical beamforming technique for an object localization operation. Adopting a neuromorphic signal processing approach on a classical microcontroller reduces the power consumption of about one order of magnitude. The efficiency of the proposed system can be attributed to the combination of the asynchronous, resistive memory-based analog circuits able to perform in-memory computing, along with the absence of analog-to-digital conversion of the sensed signal.

	Przybyla [2015][236]	Chiu [2021][237]	Jin [2014][238]	Gao [2022] [239]	This work
Application	Object localization 3D	Rangefinding 1D	Sound localization 2D	Sound localization 2D	Object localization 2D
Processing type	Beamforming	TOF estimation	Cross-correlation	Memristor-based Neuromorphic	Memristor-based Neuromorphic
Sensory system	pMUT (2 TX, 7 RX)	pMUT (1 TX, 1 RX)	3 Microphones	2 Microphones	pMUT (1 TX, 2 RX)
Localization precision	0.2^o @ 50 cm (angular)	0.63 mm @ 50 cm (distance)	1.45^o @ 1 m (angular)	9.0^o @ 1 m (angular)	10.0^o @ 50 cm (angular)
Power consumption	1.36 mW @ 100 fps	363 μW @ 100 fps	5.63 mW	30.6 μW	81.6 nW @ 100 fps

Table 3.1: Benchmarking results for the resistive memory based object localization system of this work compared to two pMUT-based systems for object localization and rangefinding, and with a neuromorphic memristor-based system for sound localization.

Comparison of our memristor-based object localization system with the state-of-the-art The precision on the target object angular position, i.e. the standard deviation of the measurements, was assessed at 10^o at 50 cm for the proposed 2D object localization system for a total system power consumption of 81.6 nW. Table 3.1 presents a comparison with previous works, namely two pMUT-based systems for object localization [236] and rangefinding [237], and a neuromorphic memristor-based system for sound localization [239]. The 3D system leverages multiple pMUT devices (2 emitters and 7 receivers) and classical frame-based signal processing to obtain the best localization precision, at the expense of orders of magnitude more power consumption for the same measurement rate [236].

Chapter 4

On-Line learning and Artificial Olfaction

The ability to adapt a neuromorphic system in situ is key for certain applications where the environmental conditions and input stimuli are ever-changing. It is the case of artificial olfaction. The particular gases to sense, humidity levels, and sensor drift over time are issues that are difficult - or impossible - to tackle with a single calibration before operating the system. In these cases, learning on-line with dedicated circuitry is necessary. This chapter introduces a system for artificial olfaction capable of on-chip learning, coupling innovative gas sensors from Aryballe - a french startup - and a RRAM-based neural network. Training is based on a hardware implementation of the Delta Rule, a simple supervised learning rule. The system is presented in detail in this chapter and a computer-in-the-loop experiment proves the capability of the proposed on-chip learning procedure. Simulations on larger-scale networks and datasets validate the concept for more challenging tasks.

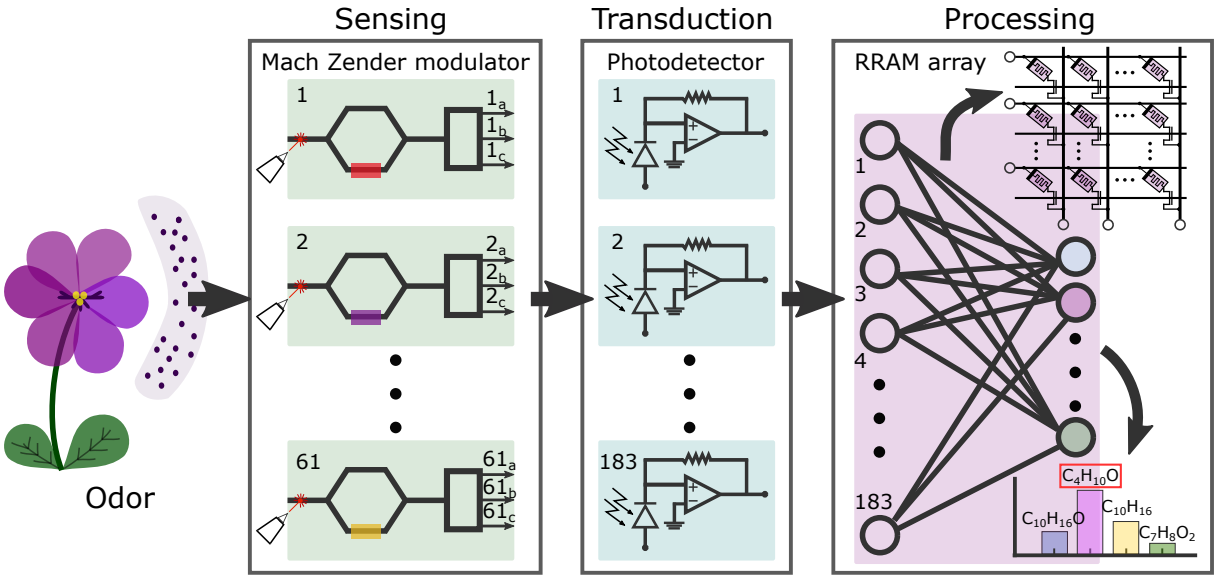
4.1 Motivation

Biological intelligence is fundamentally different from artificial intelligence (AI). One of the main reasons is the way learning occurs. In deep neural networks, learning is characterized by a training phase, where the model is confronted with a task and learns to improve at it by - mainly - gradient descent. Once the training phase terminates, the model is ready for its deployment, called inference phase. Biology operates differently: learning is not systematically scheduled and training is mixed with experiencing the context. This allows animals and humans to adapt through changing environment and to learn new tasks. While the nature of biological learning has not yet been fully understood, neuromorphic engineering and computer science have proposed methods to learn *on-line*, i.e. while the model is deployed in its operating context. The most straight-forward approach is to apply the same algorithms used for off-line training (performed on an external computer), for on-line learning too. However, this would require high computational power, going beyond the hardware constraints imposed to embedded systems. For this reason, chips performing

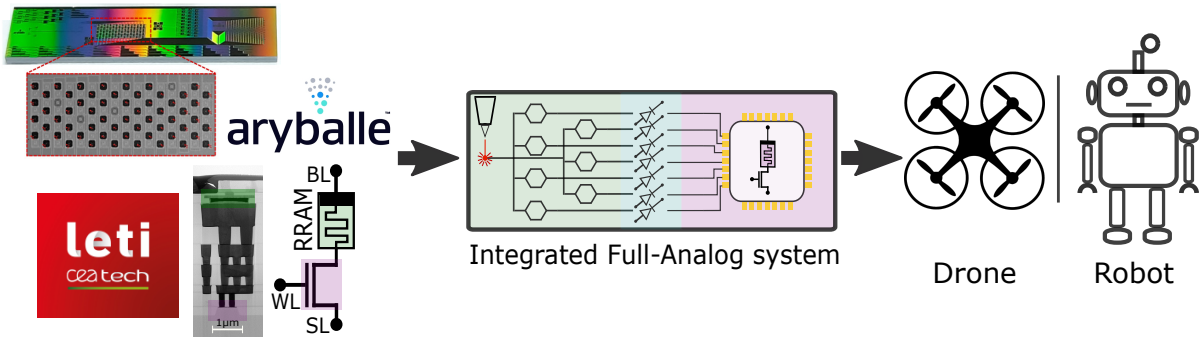
on-line learning are scarce and the general trend in on-chip learning is to simplify the training procedure to minimize the additional hardware complexity. On-chip training in neuromorphic processors mainly involves unsupervised learning rules [29, 34] or supervised learning rule with constraints to the model architecture [35]. On-line learning solutions are particularly rare for memristive systems. [230] proposes a circuit that implements the Delta Rule [240] on a RRAM-based neural network. The circuit is compatible with both artificial and spiking neural networks and trains the last layer of a neural network model. While it is arguable that such solution does not exploit the full potential of deep neural networks, the Delta Rule could represent the sweet-spot between circuit complexity and on-chip training efficacy.
To contextualize the proposition of on-chip learning for artificial olfaction, two important questions have to be answered.

Why an integrated system for olfaction? Artificial olfaction is a less developed field than vision and audition, and integrated olfaction systems that incorporate sensing and processing are rare. Conventionally, olfaction is performed with discrete sensors whose output is read and processed by a separate computer. While this scheme certainly offers flexibility and is user-friendly, it is not optimized for portability and energy consumption. However, olfactory information are important in many embedded applications: drones for fruit picking, robots navigating the environment, control systems in the food industry, to name a few. Aryballe is a french startup company in artificial olfaction presenting a complete set of gas sensing products. They envision a concept for artificial olfaction in robotics and even built a system where a robot navigates a simple environment recognizing odor cues [242]. For such applications it is desirable to minimize the size and energy consumption of the system. With reference to Figure 4.1a, this chapter proposes an embedded olfaction system featuring an array of Aryballe's Mach-Zender interferometers [59] as sensing elements, converting their information to the electronic analog domain with photodetectors and finally processing the information with a RRAM-based neural network. The integration of the sensing technology of Aryballe and the In-Memory processing, with RRAMs, of CEA Leti (Fig. 4.1b) takes artificial olfaction at the edge, targeting applications related to drones and robots.

Why on-chip learning? A legitimate question is whether on-chip learning is truly necessary. Indeed, most neuromorphic processors exploit off-line learning, where the model is trained on a computer simulation with a given dataset that models the operating conditions of the system. There are applications where this scheme fails: olfaction is a particularly interesting example. To understand the issue, one can consider the case of Aryballe's gas sensing products. When a customer purchases a sensor, the application in which the sensor will be involved in is not know *a priori*. Olfaction sensors respond differently to different gases and the response is unpredictable for an unknown gas. It is thus problematic for Aryballe to propose a model that correctly identifies the gases for the customer, without accessing their particular operating conditions. Even in the



(a)



(b)

Figure 4.1: Delta Rule on-line learning of the olfactory data (a) Artificial Olfaction with a system-on-chip where optical sensing is coupled with RRAM-based analog electronics. The system is formed by 3 main components: the Sensing part, composed of an array of chemically functionalized Mach-Zender-modulators; a Transduction layer with low-power photo-detectors; an RRAM-based classifier. (b) Aryballe's photonic sensors (image from [241]) can be integrated with CEA's RRAM-based electronics to form a system-on-a-chip that performs artificial olfaction. The end-goal is to shrink the size and energy consumption of such an olfactory system to satisfy the requirements of Edge-AI (drones or Robots).

case where the gases of interest are well known, the customer might deploy the sensor in a peculiar environment, with given humidity and gas concentration, influencing the response of the sensor. On top of that, gas sensors present some variability from one another, making the response of each specific instrument slightly unique. One strategy can be to find a computational model that is so powerful and general that it is valid in any operating condition and with any instruments. This is the approach of deep learning:

for example, computer vision models are trained on gigantic datasets featuring thousands of classes and images from different cameras [101]. However, collecting such large datasets is expensive. It follows that it is highly desirable to tailor the computational model to the specific use-case, in the exact operating conditions required for the application. This is why on-line learning is desirable for artificial olfaction.

Artificial olfaction is the perfect use-case for on-line learning with the delta rule circuit. In this chapter, an integrated system for artificial olfaction is endowed with on-line learning capabilities. First, the characteristics of an optical gas sensors are explained, then the delta rule circuit is presented. A computer-in-the-loop experiment is performed to validate the proposed concept on a 1kb RRAM array, training a neural network to classify odors from input stimuli recorded from Aryballe's sensor. At last, the delta learning rule is tested in simulation on more complex benchmarks with a Transfer Learning procedure.

4.2 Aryballe's Gas sensors

Among different other products in its lineup, Aryballe utilizes silicon photonics for gas sensing [59]. Silicon photonics is a well-established technology with various applications, including telecommunications [243] and biosensors [244]. It relies on the use of silicon or silicon-nitride waveguides, making it compatible with CMOS technology. Artificial olfaction, or the ability to detect and identify odors, involves the detection of volatile organic compounds (VOCs) using a limited number of non-specific sensors with different physical and chemical properties [245]. The global response of these sensors to a particular VOC or mixture of VOCs, known as the signature, is associated to a specific odor. Unlike analytical methods, artificial olfaction does not aim to identify the individual molecules that make up an odor or their relative concentrations. Instead, odor identification is performed using algorithms, mainly from the field of machine learning, that recognize the signature of the stimulus [60]. For example, humans can differentiate millions of different odors using nearly 400 different types of olfactory receptors [246], some at concentrations as low as a few particles per billion. Electronic noses traditionally use metal oxide semiconductors or polymers [58]. Aryballe's silicon photonics gas sensor consists of an array of Mach-Zehnder interferometers (MZI) that have been bio-functionalized with chemical receptors.

The physical principle used to detect gases is refractive index sensing. A waveguide surface is functionalized with specific receptors that bind to a particular group of molecules. When this occurs, the refractive index of the waveguide changes. Mach-Zender interferometers have been demonstrated effective at implementing this sensing principles and present easy read-out [247]. To enhance the sensitivity, the thickness of the functionalization layer can be extended so to be interacting with the electric field around the waveguide. By using porous SiO_2 layers, ethanol vapor measurement in the particle-per-billions range has been recently demonstrated [248]. However, the thicker the film, the longer the VOC will take

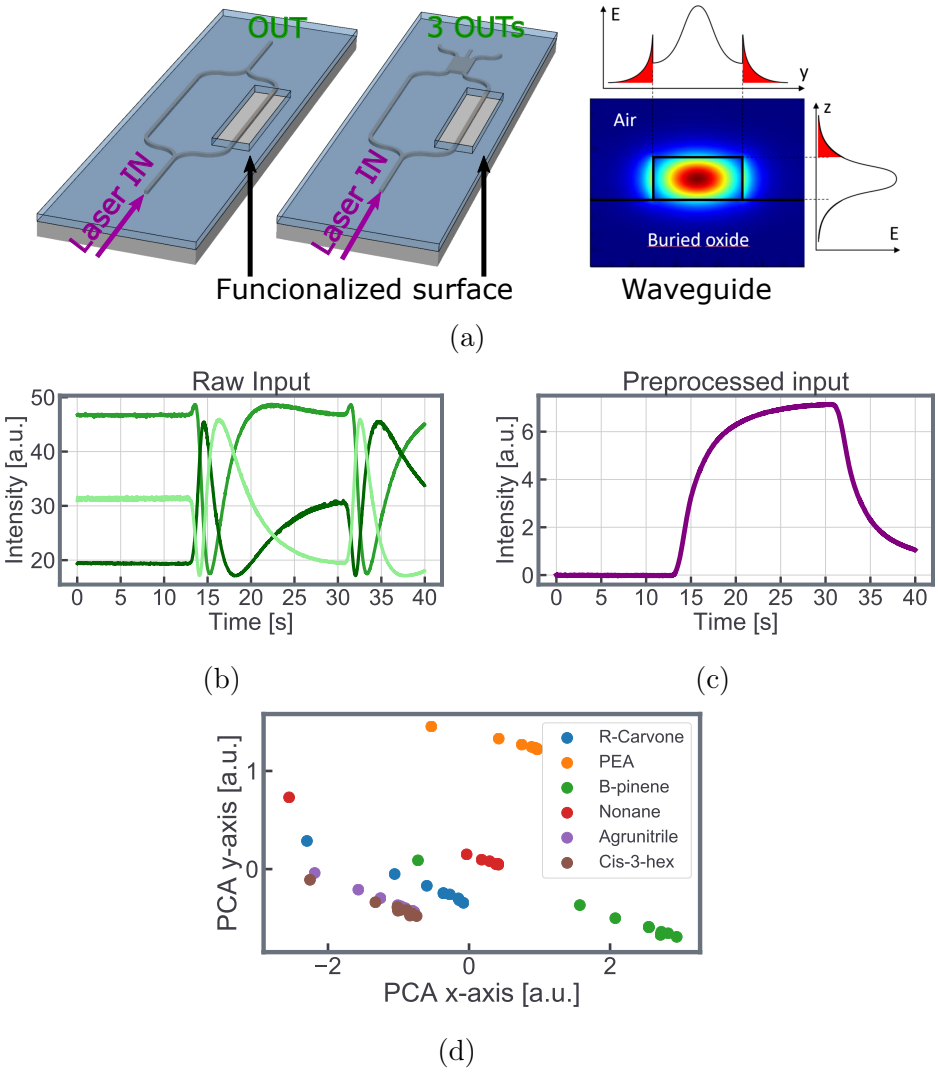


Figure 4.2: Aryballe gas sensor and its extracted information. (a) Mach-Zender interferometers (MZI) feature two waveguide, one of having a chemically functionalized surface. Light from a laser passes through both waveguides and a de-phasing is producing at the common output. To improve performance, Aryballe MZIs feature a 3-way interferometer outputting 3 light intensities. On the right, the waveguide structure with the optically confined laser light. (b) The 3 raw outputs from the functionalized mach-zender modulator, in time. An input stimulus is presented at around 15s and is removed at about 30s. (c) The 3 raw outputs from the sensors can be treated to yield the information about the intensity sensed by the Mach-Zender modulator. This preprocessing normally involves frequent sampling and conversion of the 3 outputs to the digital domain. (d) Based on the preprocessed data, Principal-Component-Analysis is performed over a dataset including 6 input stimuli. Each gas is distinguishable in the 2D PCA plane.

to diffuse down to the waveguide surface where the evanescent field is maximum. For films that are hundreds of nanometers thick, it can take tens of seconds to achieve full response, which is a limitation for real-time assessment of intermittent odor sources, tracking odors, or analyzing small volume samples. In addition, porous layers can make it more difficult to perform localized bio-functionalization. However, the solution maximizes precision in detecting gases and makes the response of the sensor easier to be classified.

4.2.1 Mach-Zender interferometer

An integrated MZI is schematically represented in Fig. 4.2a. Light from a continuous-wave laser (Laser IN) source at a fixed power and wavelength is split between the reference arm and the sensing arm with length L_r and L_s and effective index $n_{eff,r}$ and $n_{eff,s}$, respectively. The reference arm in the MZI is buried in oxide while the sensing arm is exposed to the sensing layer through an oxide-opening window. In each arm, the light accumulates a phase delay ϕ_r and ϕ_s proportional to the arm length and effective index. The initial phase has no importance, and only the phase shift $\Delta\Phi$ stores the information. The de-phasing can be expressed as:

$$\Delta\Phi = \phi_s - \phi_r = \frac{2\pi}{\lambda}(L_s n_{eff,s} - L_r n_{eff,r}) \tag{4.1}$$

where $L_{r/s}$ is the referencing or sensing branch length of the waveguide and $n_{eff,r/s}$ the refractive index at the reference or sensing branch. As gases diffuse in the environment and penetrate the functionalized layer, the de-phasing evolves along time:

$$\Delta\Phi(t) = \frac{2\pi L_s}{\lambda}\Delta n_{eff-s}(t) \tag{4.2}$$

To be detected by the MZI, gases reach the functionalized surface and diffuse in the porous SiO_2 layer. Recombining the light from the two arms results in interferences, leading to an output light intensity I_{out} varying with the phase delay difference $\Delta\Phi$ between the two arms. The relationship is as follows:

$$I_{out} \propto cos^2(\frac{\Delta\Phi}{2}) \tag{4.3}$$

To enhance the read-out, a 120^o coherent detection scheme is adopted at the output of the MZI (Fig. 4.2a). This adds two output intensities to the detectors which are experimentally demonstrated improving the average detection limit compared to a conventional read-out [249]. The detection scheme combines the reference and sensing branches with phase difference shifted by 120^o at each output. Information on the de-phasing is obtained with an algorithm computing over the evolution of the outputs along time and an arc-tangent transformation, as described in [250]. To perform such algorithm, it is customary to sample the output intensities of the MZIs with a charge-coupled-device (a rate of 30 Hz is sufficient, as described in [59]), to convert the information to the digital domain and to

process it with a micro-controller. The outputs computed by the micro-controller are said to be the *Preprocessed* data. We propose to avoid this algorithm and to work directly with the output powers of the MZI devices. After translating the optical power information to the electronic domain with a photo-detector, the information is available as a voltage, this constituting the *Raw* data. This approach minimizes the pre-treatment of the sensed information and is in the spirit of energy-efficient systems. However, taking the proposed approach suffers from some limitations which are explained below.

Limitations of this approach Directly utilizing the Raw data and avoiding sampling the signal to extract information is certainly adherent to the low-power spirit of Edge AI. Its field of validation is however constrained to low de-phasing $\Delta\Phi$. When the initial state of the outputs from the MZI is known and the variation of the phase is less that 360^o, the relative de-phasing can be uniquely decoded from the three MZI Raw outputs. However, when the de-phasing is equal to 360^o, the Raw output returns to the initial state and the information on a further de-phasing is lost. For such cases, the classical pre-processing technique explained in [250] is necessary to extract the temporal evolution of the gas concentration. To avoid large de-phasing of the sensed branch one can act on the functionalized layer to decrease the surface sensitivity. Alternatively, one can limit the analysis to low gas concentrations so that the de-phasing results small enough not to overcome the 360^o barrier.

4.2.2 Collecting a dataset

An Aryballe instrument for gas sensing is used to collect a dataset. The reference instrument is called NeOse [251] and it features 64 MZIs functionalized with different chemicals. 6 instruments are involved in the dataset, each built with the same process and functionalized with the same chemicals. In theory, the instruments are all responding in the same way to the same stimuli, however process variations in the MZIs makes each instrument slightly unique. The 6 sensors are exposed to 7 gases in a controlled environment, a chamber where the chemicals are released and flushed out. Each gas is presented for a period of about 20 s and a 15 s interval is interleaved between measurements. In each session, the gases are recorded for 8 times each, for a total of 56 data-point per session. 5 sessions per instrument are repeated thus bringing the data-point count to a total of 1680. To acquire the information, MZI's three outputs are captured by Charge Couple Device pixels and converted to the digital domain, where the data is sampled at 30 Hz. The response of the MZI's three outputs in instrument 1 is plot along time in Fig. 4.2b. The three green traces correspond to the *Raw* data from the sensor. As it can be seen, they oscillate from about 14 s - when a gas is allowed in the chamber - and start reverting back to the initial state at 32 s, once the gas is flushed out. The information from the three outputs can be treated to yield the *Pre-processed* data, shown in Fig. 4.2c. The oscillations of the 3 output branches are combined to sense the intensity of the response of the MZI. As for the Raw data, the Pre-processed data is activated at 14 s and the

stimulus is sensed up to the 32 s mark. The MZI then returns to its initial condition. To assess the efficacy of the sensing system, the *Pre-Processed* data from instrument 1 is analysed with a Principal-Component-Analysis (PCA), Fig. 4.2d. The seven gases result well separated in the first two principal components. This is a testament to the quality of the sensing instrument, which makes the input olfactory information linearly separable and thus easy to classify. Similar results are obtained with all the instruments in the dataset. However, a drift in each of the clusters is visible, due to the temporal drift of the base-line activation of the sensor. This phenomenon is mainly due to temperature variations in the instrument: a thorough experimental evaluation of the response of the sensor depending on the temperature is carried out in [241]. A first analysis of the dataset involves software model of classifiers, to investigate how well the registered olfactory cues can be distinguished. Software logistic regression algorithm is performed on portions of the datasets: when possible, the subsets are split into 50% for training and 50% for testing. Results are reported in Table. 4.2.2. Two sets of experiments are performed: in the first case, the training and testing occurs with data from the same instrument, considering one instrument only; in a second run, all the instruments are considered and 5 of them are used as the training set, the remaining one as the test set. For the first case, classification results are generally very high, 100% for *Pre-processed* data and an average 97.07% for the *Raw* data. This means that when training on the same instrument in which one performs inference, results are always satisfactory. However, performance change when training and testing involve different instruments and specifically when the test instrument is not included in the train set. On the right side of the table, training occurs on all the instruments except for the one indexed by the number in the column. For the *Pre-processed* data, performance oscillates depending on which instrument the algorithm is tested on: data from instrument 6 is still easy to classify, while instrument 2 falls below 30% of accuracy. This is due to the variability in the response of each instrument which slightly changes the signature of the gases. Classification accuracy on *Raw* data is at chance-level ($\tilde{1}6\%$) for all instruments. The reasons for such bad performance could be that device-to-device variation between instruments impact even stronger on the unprocessed inputs. These results reinforce the message that on-line learning is important to artificial olfaction. While considering a wide dataset with multiple instruments is not helpful when testing on a different instrument (right column of Table. 4.2.2), it suffices to add data to the train set from the same instrument performing inference to recover good performance (left column of Table. 4.2.2).

4.3 The Delta Rule for RRAMs

The Delta Rule is a simple algorithm to learn in a supervised manner on a single layer neural network [240]. The algorithm performs gradient descent on a perceptron network, and can be considered as a special case of the back-propagation algorithm [17]. The delta rule minimizes the mean squared error (MSE) of the perceptron respect to its target outputs: $MSE = \sum_i y_i^2 - t_i^2$, where t_i, y_i are the target and output at the i^{th}

Training and testing on the same Instrument		
Train Instrument ID	Test Accuracy [%]	
	Processed	Raw
1	100.00	97.16
2	100.00	88.90
3	100.00	99.48
4	100.00	98.06
5	100.00	98.97
6	100.00	99.87

Training on all Instruments but one, testing on the remaining Instrument		
Test Instrument ID	Test Accuracy [%]	
	Processed	Raw
1	83.55	14.84
2	76.45	16.77
3	29.03	9.03
4	92.90	9.35
5	90.00	14.19
6	100.00	13.87

Table 4.1: Classification accuracy on the Aryballe olfactory dataset, based on two partitions of the data. On the left, the training and testing sets include one instrument only. On the right, training involves all instruments but one, the remaining one being used as the test set. Green color highlights good results on *Raw* data, while the red color is used in case of bad accuracy.

dimension. It does so acting on the weights W, with a weight update that can be derived as $\Delta W = \mu(T - Y)X$, where μ is the learning rate, X is the n-dimensional input, Y and T are the m-dimensional output and target. In classification, the target is encoded as the activation of a specific output neuron, while the rest is set at 0.

4.3.1 From the algorithm to the circuit

This algorithm was proposed in [230] as an effective way to perform on-line learning for in-memory-computing (IMC) architectures. The delta rule is perhaps a good compromise between effectiveness of the on-line learning and its complexity added to the architecture. Implementing the back-propagation algorithm on-line requires to store into memory the activation of all the neurons in the model and to compute matrix-vector-multiplications involving the transpose of the weight matrices [17]. On top of this, back-propagation is problematic in IMC architectures as one has to deal with low-bit precision and noisy devices. IBM has embarked in this challenge proposing a 1T1C2R device, a capacitor next to two PCM devices, with the capacitor acting as an analog volatile memory. The capacitor integrates the weight updates and thus increases the combined capacitor+PCM bit precision while learning [131]. This approach suffers from the large area overhead of the capacitor that is needed per synapse. Another approach on in-situ learning is proposed by Dalgaty et al. in [134]. The bayesian learning algorithm Markov-Chain-Monte-Carlo (MCMC) is compatible with most NVM, as it exploits the inherent cycle-to-cycle and device-to-device variability to explore the parameter space and converge to a collection of well performing models in the posterior distribution. However, the MCMC algorithm is very slow to converge, meaning it is adapted to small networks and easy tasks. Furthermore, MCMC requires multiple devices per synapse, resulting in large memory footprint and area consumption. The delta rule approach [230] combines the effectiveness of gradient-descent

learning, making use of 2 devices per synapse, and mitigating the variability issue of NVMs with the learning procedure it-self. In particular, [230] focuses on a RRAM-based implementation, even though the learning scheme is applicable to any Non-Volatile-Memory, except for the programming circuitry which has to be device-specific. The learning algorithm can be summarized in Algorithm 3. Weights, implemented by the RRAM array, are initialized with a SET operation. With a differential architecture of the array, the weights are obtained by subtracting positive and negative conductances. The latter are obtained with a negative reference voltage applied at the sense amplifier reading the array column. A scaling factor s_{WG} converts conductance to the weight matrix. At every input presented to the array corresponds an output target. The delta rule circuit computes the weight update accordingly, and performs the update on the RRAM devices with a probability p_{prog}. A weight update occurs by a first RESET operation and then a SET operation with V_{SET} calculated by the delta-rule circuit. The weight update is avoided when the error $(Y-T)$ is lower than a certain threshold ΔY, this helping the stability of the algorithm.

Algorithm 3: Delta Rule algorithm

$G_0 \leftarrow SET(V_{SET,init})$
$W_{out} = (G_0^+ - G_0^-)s_{WG}$
s_{WG}: scaling factor between weight and conductance
p_{prog} : probability to make weight update
Δy: error threshold
μ : learning rate
for X, Y in train_loader:
    # X: input, T: target output
    Y = X $\times W_{out}$
    $\Delta W_{out} = \mu X \times (T - Y)$ - [Delta Rule circuit]
    $V_{SET} \leftarrow \Delta W_{out}$ [Delta Rule circuit]
    With probability p_prog:
        # perform weight update on RRAM
        RESET selected devices
        SET selected devices with V_{SET}
        $G \leftarrow SET(V_{SET})$
        $W_{out} = (G_0^+ - G_0^-)s_{WG}$
    end
end

The algorithm is also compatible with Spiking-Neural-Networks with an additional circuit that enables the delta-rule circuit at the output neuron's spike. The learning rule has been demonstrated on the MNIST dataset, with a simulation calibrated on experimental data, performing at 92.68%. The probability of programming the RRAMs in the weight

update (P_{prog}) revealed to be an important parameter of the learning scheme. In fact, the variability in the High-Conductance-State makes low weight-update ineffective, especially where the update is lower than the standard deviation of the device variability. This is why the RRAM-compatible delta rule operates with high learning rate (μ). To avoid sharp variations of the weights, the probability of weight updates (P_{prog}) assures that such important variations of the weight only occur at a fraction of the total devices. This reduces the effect of a large learning rate, while still assuring the descent of the gradient.

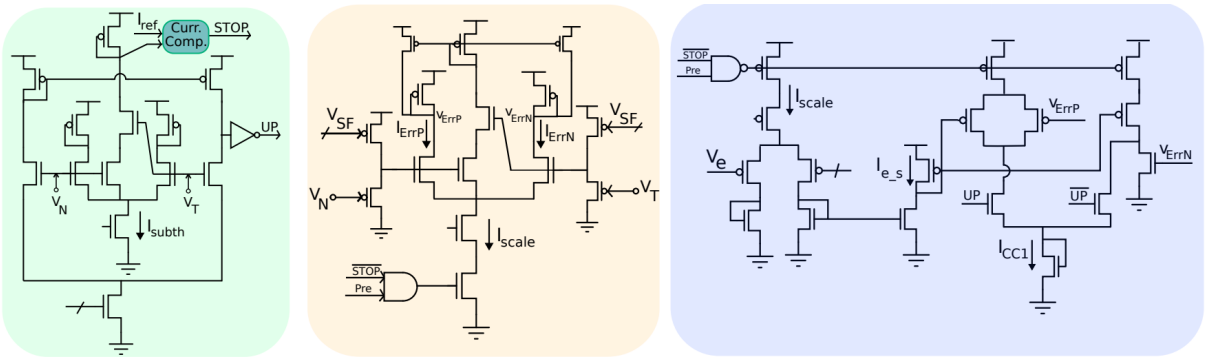


Figure 4.3: Delta Rule circuit, from [230]. The circuit is divided into three blocks. The green one is a Bump circuit taking V_N (the output of the neuron) and V_T (the target for that neuron) and determining the direction of the weight update (UP or DW), or engaging the STOP signal when no update is to be made. The yellow block is a scaling circuit to normalize the different between V_N and V_T and produce a scaled current I_{scale}. The blue block is a second scaling section, where the current is controlled by the bias V_e, modulating the compliance current for the weight update I_{CC}

At the heart of the delta rule implementation, is the delta rule circuit (Fig. 4.3). Considering a neuron in the output layer, this will output a voltage V_N and will have a target activation expressed by the voltage V_T. These voltages are subtracted using a subthreshold "Bump" (subBump) circuit [252] highlighted in green, and an above-threshold "Bump" circuit (abvBump) in orange. The subBump circuit compares V_N and V_T giving rise to the error currents when neuron and the target frequencies are far apart and generates the STOP signal when the error is small and in the stop-learning range (δ_{th}) [253, 231]. STOP signal gates the tail current of all the above-threshold circuits and thus substantially reduces the power consumption when the learning is stopped. The abvBump circuit subtracts V_N and V_T and scales it to Iscale, equal to the maximum I_{CC} required to SET a RRAM device. Based on the error sign (UP), the scaled error current is summed with or subtracted from the scaled device current generating the desired I_{CC}. This circuit is highlighted in purple. The compliance current can be converted to the gate programming voltage V_{SET} with a simple diode (not shown in the schematics).

The delta-rule circuit is embedded in a conventional RRAM array as shown in Figure 4.4a. An RRAM array is designed with the differential configuration, where source lines are

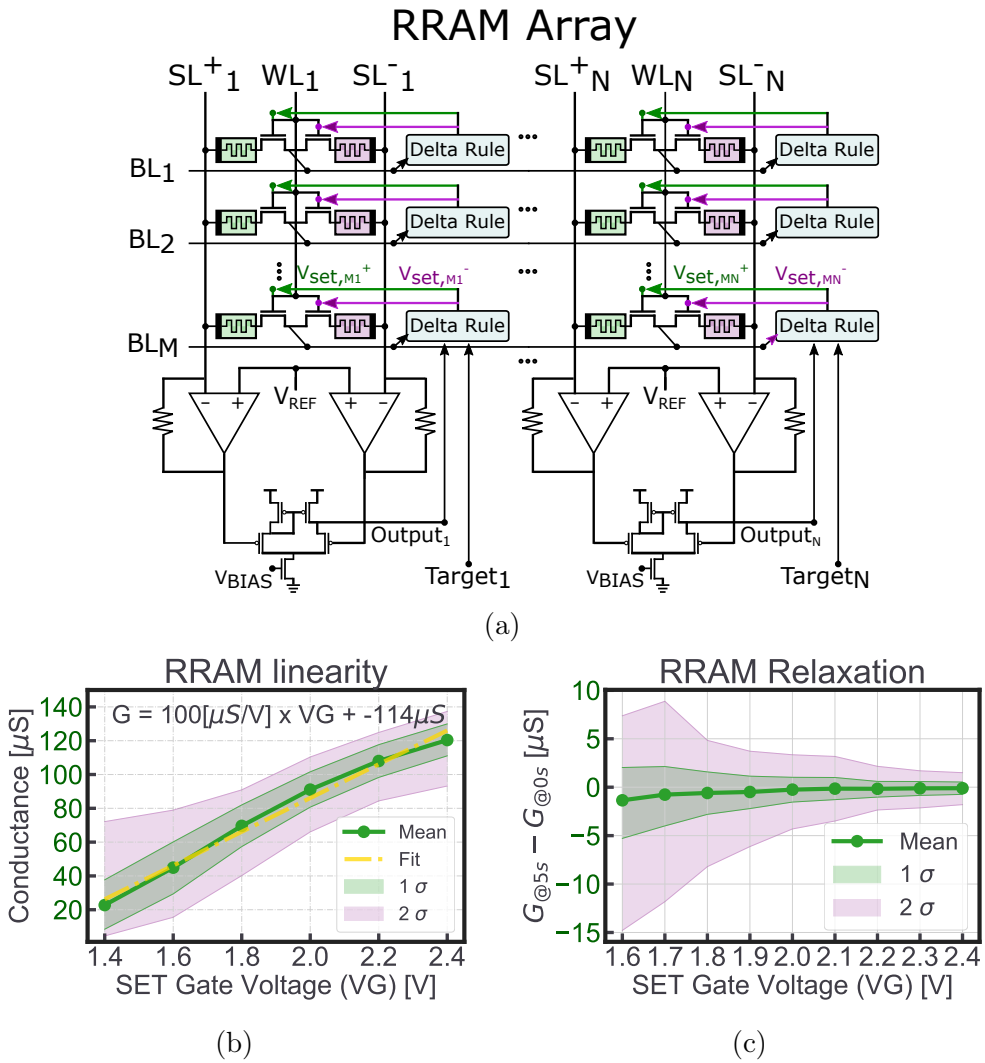
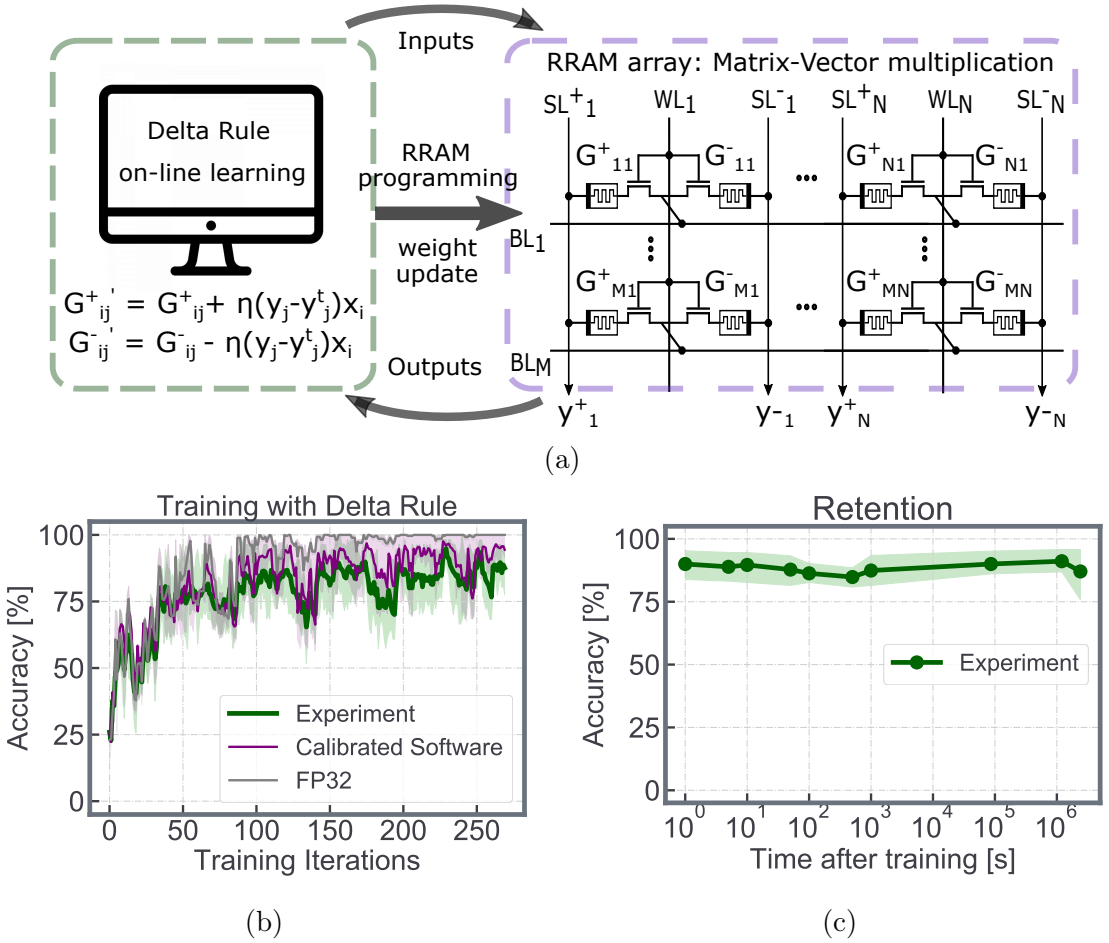


Figure 4.4: RRAM array featuring the Delta Rule circuit. (a) Schematics of the array. Inputs are fed by the bit lines and source lines carry the output current implementing the MAC operation. The output from each column is compared to a target value and scaled by the input (BL_n), to yield a SET gate voltage ($V_{SET,mn}$) - with the Delta Rule circuit - for each device's weight update. (b) RRAMs characteristics depending on the gate SET voltage (VG). The plot shows the linear behavior of the RRAMs conductance after the SET operation, highlighted by the yellow line and the linear equation on the upper part of the figure. First and second standard deviation reporting the variability of RRAMs are shown with the shaded areas around the mean conductance value. (c) RRAMs' variability is also in the temporal domain, in the form of Relaxation. The array's conductance is plot after programming (t=0s) and after 5 seconds (t=5s) for different gate voltages (VG). While the mean conductance is stable, devices tend to change conductance, especially when programmed with weak VG.

biased with either positive or negative voltages when reading the array. Inputs are provided to the array by means of voltages applied to the bit lines (BL). Currents sum at the source lines (SL) performing the multiply-and-accumulate operation, and are read down the columns with a sense amplifiers. The outputs are obtained by subtracting the currents from the positive and negative columns of the array. The outputs are first compared to their targets and then scaled by the inputs by the Delta Rule circuit. This generates the set gate voltage ($V_{SET,\mp}$) for the positive (green) and negative (violet) devices. To correctly generate a weight update that obeys to Equation [reference], it is required that the devices respond linearly to the programming operating conditions. Different studies analysed non-linear devices for on-line learning trying to cope with non-linear weight updates [131, 254]. In RRAMs, linearity appears when setting the device with a control over the gate voltage V_{SET} or VG. It is recalled that progressive SET operations are not effective on RRAMs as they are for PCM device [255, 256], so an intermediate RESET operation is required to gain control over the conductance with a SET operation. To analyse the linearity of RRAMs, different SET with varying gate voltages are applied to a 1kb array, measuring the resulting conductances at each device. A linear behavior appears in Fig 4.4b as a function of VG, highlighted by the yellow linear interpolation. The coefficients of the interpolation are reported in the equation on top of the plot. Variation of conductance characterizes RRAMs' high-conductance-state as shown in the Fig 4.4b by the shaded areas. The standard deviation, especially the second degree, gets larger as the SET gate voltage is smaller, meaning that stronger SET operation result in tighter distribution of conductance. RRAMs also present temporal variability. Relaxation is the most critical form for on-line learning: a fast phenomenon manifesting as the settling of the conductive filament of the RRAM cell after a SET operation. To quantify relaxation, a measurement has been performed on the same 1kb array where all the devices have been SET and the conductance are immediately read. After 5 s the conductances are read again and the values are compared to the initial ones. The experiment is repeated for SET at different gate voltages. Difference in conductance over time is plot in Fig. 4.4c. The mean value of the difference is in dark-green and shows that the mean values do not change over time, except for the lowest gate voltages: in these cases, it is probable that part of the devices have their conductive filament partially disrupted. The first and second standard deviations are shown with the green and violet shaded areas. Notably, the spread of the distribution changes with the gate voltage, meaning that device SET with higher voltages are more stable over time.
These measurements frame RRAMs among the most adapted devices for on-line learning thanks to the linearity of the SET operation and the temporal stability of the high-conductive-state.

4.3.2 Computer-in-the-Loop experiments

The vision of the project is to combine an RRAM-based network endowed with on-line learning capabilities to the Aryballe's gas sensors, so to perform area and energy efficient



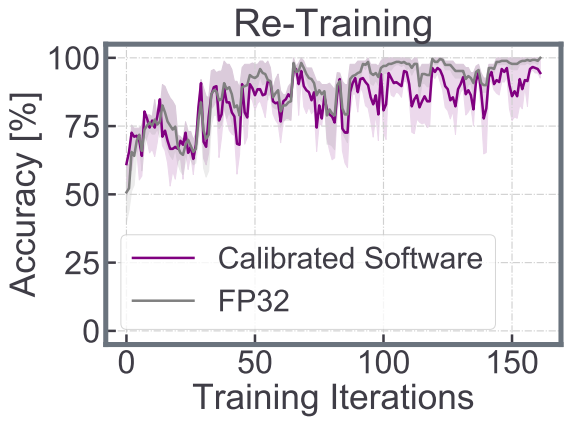
(a) (b) (c)

Figure 4.5: Delta Rule on-line learning of the olfactory data (a) Sketch of the Computer-In-The-Loop experiment. The computer inputs the dataset to the RRAM-array, via the bit lines. The array performs the MVM operation yielding the results through the source line. Output currents are read by the computer which then calculates the weight update. The weight update is performed by programming the RRAM devices with the appropriate conditions. (b) On Line learning with the Delta Rule yields good classification accuracy on RRAM-based network. The reference is the FP32-based model which reaches 100% after 200 iterations. The software calibrated model of the RRAM network (purple) shows good performance too, despite the instability due to the C2C and relaxation variability of RRAMs. The computer-in-the-loop experiment (green) confirms both the problems related with learning with RRAMs, despite showing that good accuracy can still be achieved after 250 iterations. (c) After training with the computer-in-the-loop experiment, the RRAM model is repeatedly read to measure the accuracy over the course of time. Even after 1 month, relaxation has not broken the RRAM model, which only suffering from a minor accuracy decrement.

artificial olfaction. To validate the concept, a computer-in-the-loop experiment has been performed. A reduced version of the dataset recorded by Aryballe is formed selecting

data from Instrument 6 and considering four input gases only. The choice of limiting the data to a single instrument matches the user experience of a customer buying a specific Aryballe gas sensor. The dataset features 255 data-points, among which 51 are chosen as the training set and the remaining 204 as the test set. Each data point features 183 inputs, from the grid of 61 mach-zender interferometers. The model of choice to classify the gases in the dataset is a logistic regression algorithm, performed with a RRAM-based perceptron. The IMC hardware platform to implement this algorithm is a 32x32 RRAM array with a periphery designed for both per-device programming and parallel reading of the matrix. This allows performing the multiply-and-accumulate operation on the chip. However, a differential architecture for the device is required by the algorithm to express both positive and negative weights: to do this, the array is split in positive and negative columns. During the parallel reading operation, the array is read column-wise and subtraction between positive and negative values of conductance is performed off-chip, on a computer. To fit the input data to the 1024 device array, the input is down-sampled from a size of 183 to 128. This is carried out by selecting the most significant input channels with a software logistic regression algorithm: the 128 input channels with resulting weights with highest absolute weight values are selected. Accounting for 2 devices per weight, and with an output layer of 4 neurons (4 gas types), the device count amounts to $128 \times 2 \times 4 = 1024$, thus saturating the 32x32 array.
The computer-in-the-loop experiment interfaces an Arduino micro-controller and a Keysight B1500 [257] machine to the RRAM array: the micro-controller is commanded by a python file run on an external computer and executes the addressing of the array; at the same time, the computer controls the B1500, which is devoted to programming and reading the RRAMs. More information on the setup are described in Appendix 7. A simplified scheme of the experiment is shown in Figure 4.5a. Inputs - from the reduces Aryballe dataset - are fed to the array in the form of voltages. The RRAM array performs the MAC operation column-wise yielding the output currents read by the B1500 and stored on the computer. Here, weight updates are calculated with the delta rule algorithm, preparing for the programming phase: the micro-controller addresses the appropriate cells and the B1500 performs the RESET and SET operations with the calculated V_{SET}. The cycle - said iteration - begins again and repeats for each input in the dataset.

Results from the Computer-in-the-loop experiments are shown in Figure 4.5b where the training procedure lasts for 254 iterations. Along with the experimental results (green), the test reports the results for a simulation based on hardware data (Calibrated software, violet) and with floating-point 32-bits (FP32) weights in a pure software model (grey). In the calibrated software case, RRAM characteristics from Fig.4.4b and Fig. 4.4c are fit and included in the simulation. Notably, the dataset is learned within few hundred iterations, demonstrating the efficacy of the sensor array at separating the gas cues into easily classifiable odors. In fact, the test accuracy for the FP32 model saturates at 99% after 150 iterations. The cycle-to-cycle variability and relaxation effect introduced by



(a)

Figure 4.6: Delta Rule re-training of an model with degraded performance. (a) The Delta Rule on-line learning procedure can be deployed for an model requiring an update. An ideal FP32 model is capable to recover good performance fully in a few hundred iterations. Also a RRAM-based model (in purple, in the Calibrate Software experiment) is retrained with the same speed, and reaches almost the same accuracy in the olfactory dataset classification task.

RRAMs inevitably reduce the performance of the algorithm: however, the experiment reaches 90% of classification accuracy on the test set. This is in agreement with the software model calibrated on the hardware data. Fluctuations of accuracy are partly due to the nature of stochastic gradient descent (SGD) and partly to variability of RRAMs. Showing a single data-point per time to the model, and with a large learning rate imposed by the use of RRAM, means that the model's parameters oscillate at each iteration, changing the conductances and weight values abruptly. However, the delta rule training still converges and the model classifies the inputs with 90.56% mean accuracy over 10 experimental runs. After training with the delta-rule, the RRAM array is left untouched, and is then read again multiple times over the course of a month. This tests the Retention of the accuracy over time. RRAMs are known not only for Relaxation as temporal variability, but also for longer time-scale instability. Recent studies on RRAMs for Edge-AI have shown that RRAM-based models suffer from Retention [225, 258] but that is not breaking the performance of neural networks [129, 94]. For the 10 experimental models, accuracy over 1 month period is shown in Fig. 4.5c. The plot highlights a slight decrease of performance after 1 month, that being of 2.78%. However, the decrease is not incremental in time and evidence shows that a total collapse of accuracy in the models over longer periods is not expected.

Degradation of performance is certainly not desirable and has to be minimized, but it is not catastrophic to a model endowed with on-line learning capabilities. In fact, when

the performance is not satisfactory, one can resort back to the training phase and recover performance. To demonstrate the concept, a Re-Training procedure is performed on the calibrated software and FP32 models over a validation set of 51 data-points, different from the training set with which the models were originally trained (Fig. 4.6a). 10 trained models are added a gaussian noise of 20% of standard deviation respect to the maximal weight absolute values. This decreases the accuracy from about 95% down to around 60% and emulates a heavy disturbance in the RRAM array. Despite the large decrement of accuracy, the original level of performance is restored after 100 iterations and consolidated after 150 iterations.

4.4 Transfer Learning with Delta Rule

On Line learning is a very desirable ability for Edge AI systems, but it introduces complexity at the hardware level. This is why, in practice, on-line learning is still mostly unexplored, especially in commercially available solutions. The reason is that learning at the edge requires additional circuits which - in some cases - defeat the purpose of maximally efficient implementations of neural networks. The most common algorithm to train neural networks is gradient descent with the back-propagation technique. With this method, the activations of all neurons have to be stored to memory and the weight matrices are involved in the computation of the gradients. To get around such an expensive learning algorithm, several alternatives have been proposed, like RTRL [114], e-prop [115], eq-prop [116]. While these algorithms all relax the hardware requirements respect to the original back-propagation algorithm, they introduce additional complexity and challenges for designers: for example, RTRL increments the complexity of training to $O(n^4)$, e-prop requires a state variable with long time constant per synapse, eq-prop transforms a conventional artificial neural network to a dynamical system and introduces bidirectional synapses.
This is why the most conventional method for on-line learning is based on Transfer Learning. Transfer learning is the research problem in machine learning focusing on gaining knowledge from a task and porting that knowledge over to solve a new - but related - task. It is customary that the original task for transfer learning is "general" enough that the particular task which the model is later asked to perform will result easier to solve. This scheme is particular effective for Edge-AI, in which the power of a model trained off-line on a server can be specialized in the context of a particular application at edge. For example, a vision model classifying over thousand of classes of images can be used on a security camera to recognize intruders. Specializing the model to the edge application still requires on-line learning abilities, but is less demanding than training the model from scratch. However, the question of how to fine-tune the model on-line remains open.
We propose to simplify the transfer learning pipeline and restricting learning to the last layer of the neural network, as in the Delta Rule algorithm [240]. While this might sound simplistic, it can potentially lead to surprisingly good results. The Delta Rule learning fits well within the Transfer Learning framework, where one exploit the power of a large neural network model previously trained on a large dataset. By learning on the last layer

only, one combines the computational power of a large model with the simplicity of a linear classifier. Moreover, this works with any neural network architecture and task. The concept is summarized in Figure 4.7a. The general neural network model has parameters W_{hid}, the weights of the hidden layers. These are trained on a large, general training set, making the model very powerful. However, the network might have to be specialized for a particular set of data, collected on-line. In this case, one learns the last layer only with the Delta Rule, as highlighted in Fig 4.7a.
In our case, we are interested to perform the Delta Rule algorithm on RRAMs so the focus is on whether the devices' variability and non-idealities hinder the network to learn the data correctly. In the rest of the section, the performance of the Delta Rule algorithm on RRAMs in the context of transfer learning is thoroughly analysed, particularly focusing on what is the variability level allowing good performance on the MNIST and CIFAR10 tasks. Two different networks are used for the two tasks: a Multi-Layer-Perceptron (MLP) and a Convolutional-Neural-Network (CNN).

MNIST dataset The MNIST dataset is the basic benchmark in computer vision. In this experiment, the training dataset is divided in 2 parts: 95% of it is used as the training set and 5% as validation set (3000 images), or second training set. A Multi-Layer-Perceptron (MLP) is chosen to classify the hand-written digits, having 2 layers of 256 neurons each. The training architecture is first fully trained with conventional back-propagation using the Adam optimizer [259]. This allows the network to achieve a classification accuracy of 97.99%. After this first training phase, the output layer is replaced with a new one trained with the Delta Rule on a software model calibrated on RRAM data. The validation set is used in the delta rule learning phase. While this is certainly not a realistic use-case of transfer learning, it is a good benchmark for the RRAM-based delta rule on-line learning procedure.

Figure 4.7b shows the evolution of the classification accuracy spanning over the 3000 iterations constituting the validation set. Two different models have been proposed, one featuring all kinds of RRAMs variability effects (violet) and one without Relaxation (green). Interestingly, Relaxations seems to have a large impact to the results, making accuracy violently oscillate along the training iterations. This is a symptom of the instability of the classifier to the variations of RRAMs, especially for those devices that are programmed with a weaker SET operation. When Relaxation is not accounted in the simulation, performance is still not on-par with the FP32 model, but it converges to 96.06% with an accuracy loss of less than 2%. This is surprisingly positive considering the cycle-to-cycle variability of the devices. Furthermore, the classifier is trained over a small subset of the dataset and the hidden weight exploited the knowledge acquired during the first training phase and are not modified further. RRAM's Relaxation is certainly a deal-breaker in this experiment, however it has been shown that it can be traded off with latency in the programming operation to make RRAMs more stable [260]. As mentioned before, the

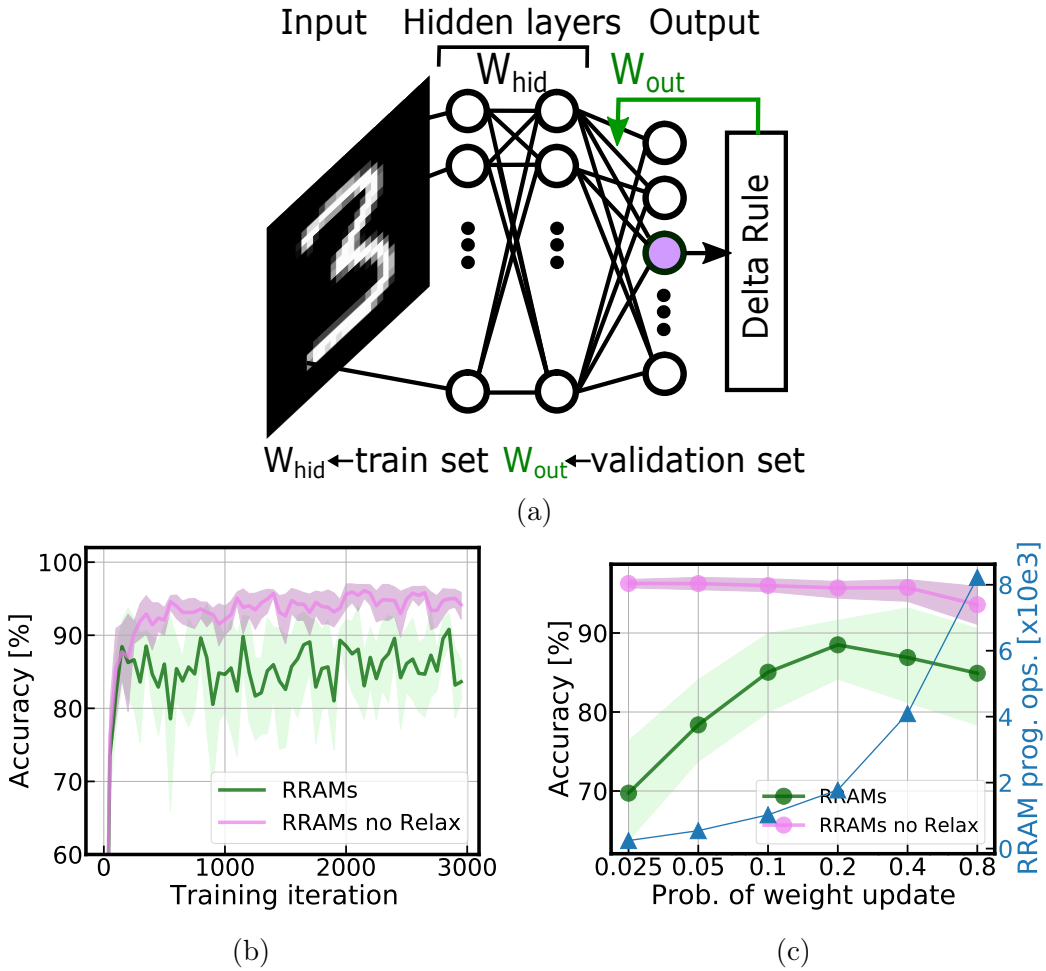
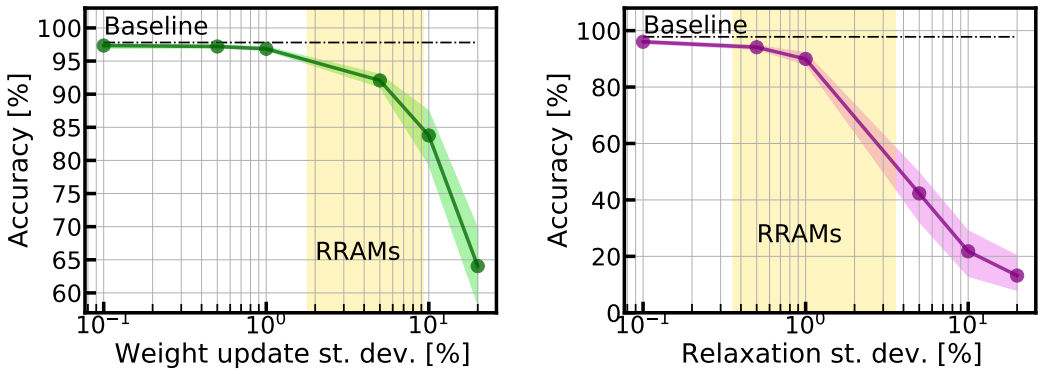


Figure 4.7: Transfer Learning with the Delta Rule. (a) A MLP network with 2 hidden layers is used to classify handwritten digits (MNIST dataset). The network is fully trained on FP32 weights and those are later converted to RRAMs. The hidden weights W_{hid} are frozen after training on the training set. Output weights W_{out} are trained from scratch on the validation set with the Delta Rule algorithm, directly on the RRAMs. (b) The Delta Rule transfer training is performed either enabling or disabling the Relaxation effect of RRAM. Including Relaxation (green) makes the learning unstable and with lower accuracy. Removing Relaxation (purple) stabilizes the training procedure and allows reaching higher accuracy. (c) The weight update probability (p_prog) is an important parameter in the Delta Rule algorithm. When relaxation is disabled, it has little effect, though lower p_prog yields better results. When relaxation is enabled (green), higher p_prog balances the chaotic effect of RRAM temporal variability. The weight update probability is related to the total number of programming operations (prog. ops.) as shown in the blue plot.

delta rule algorithm features a probability in making a weight update on the RRAMs. This is important to deal with the variability of RRAMs which results in the need of high learning rates. The programming probability P_{prog} mitigates the effect of the large learning



(a) (b)

Figure 4.8: Variability analysis learning MNIST online. (a) In this test, the weight update was performed adding random noise. Noise is injected from a Gaussian distribution with given standard deviation (x-axis). More noise of course upsets the accuracy of the mode, as observed in the plot. The Delta Rule learning tolerates noise up 5% of standard deviation. The yellow bar represents the noise levels of RRAMs. (b) Relaxation is the short-term temporal variability of RRAMs. Here, it has been introduced as a Gaussian noise in the weights, between any two training iterations. High level of relaxation destroys the ability to learn the task. The yellow bar shows the relaxation values of RRAMs.

rates and makes the training smoother. This is plotted in Fig 4.7c where accuracy after 1 epoch of training is analysed as a function of P_{prog}. Again, RRAM relaxation (green) decreases the performance by a large margin respect to the case where only cycle-to-cycle is accounted for (violet). For the latter case, P_{prog} is not as effective, even though performance decreases with large programming probability, as the large learning rate and variability of RRAM weights become problematic. When Relaxation is accounted in the RRAM's model, the probability of re-programming becomes much more relevant. As the temporal variability hits most of the devices, not just the ones that have been re-programmed, lowering P_{prog} makes the network unstable as most weights move their values without being updated by the algorithm. On the other hand, when P_{prog} is too high, the effect of the cycle-to-cycle variability becomes too aggressive. A compromise is found at $P_{prog} = 0.2$. On the right hand-side of Fig 4.7c is the number of programming operations performed in each simulation. Since the programming count is proportional to P_{prog}, the blue curve is linear with the x-axis. This goes to show the importance of keeping P_{prog} low, so that less energy is invested in the training operation.

In a second round of experiments, the robustness of the delta rule algorithm is tested by replacing the RRAM variability model with simple gaussian noise with tunable standard deviation. Results are reported in Fig. 4.8a and Fig. 4.8b. First, gaussian noise is applied to the weight update, recalling the cycle-to-cycle variability of RRAM. Fig. 4.8a plots the accuracy in the test set as a function of the weight update noise, showing that - as intuitive

- variability in weights is detrimental to performance. The horizontal bar reminds the 97.99% accuracy baseline when no variability in the weight update is presented. A yellow bar shows the variability level of RRAMs and it is evident that RRAMs' cycle-to-cycle is at the edge of breaking the performance of the algorithm, but still allow for good performance (up to 96%). Increasing the noise to the 10% mark lowers the performance considerably (84.3%). Next, weight update noise is removed and the effect of the relaxation is modelled in the form of a gaussian noise added to all the devices. Fig.4.8b plots the accuracy depending on the relaxation noise amplitude. As relaxation involves all the devices, it is thus stronger and breaks performance earlier than the weight-update noise. RRAM's relaxation noise range is shown again with a yellow bar. Relaxation is a more dangerous and effective non-ideality for devices implementing synaptic weights and it has to be reduced. As already mentioned, different programming techniques [260] and material stacks have been proposed to mitigate relaxation in RRAMs.

CIFAR10 dataset The Delta Rule is also tested with a more complex task, classifying the CIFAR10 dataset. To do so, a Multi-Layer-Perceptron would limit classification accuracy, so a more powerful Convolutional Neural Network is chosen as the pre-trained model. Thanks to its favorable compromise between classification accuracy on the Imagenet dataset [101] and the small memory footprint, EfficientNet_B0 [261] is used as the support model for the transfer learning task. The network features 5.3 Mb parameters and requires 390M floating-point operations per inference. A sketch of a CNN is shown in Fig. 4.9a: an input image is processed with multiple convolution and max-pooling layers, the output of the CNN is then flattened and passed to a linear classifier, trained with the Delta Rule algorithm.

CIFAR10 images are scaled to 224x224 pixels and processed with EfficientNet. The delta rule algorithm is used to train the output layer. Due to the increased complexity in classifying the dataset, training with a batch size of one is more difficult. Preliminary tests conducted with different batch-size during training reveal that a batch-size of 1 is insufficient and satisfactory classification accuracy (>75%) is reached with a batch size greater than 8 (See Appendix 7). For the following experiments, batch size is fixed at 32. To train on-line with batch-size greater than 1, one would have to store outputs and activations of the penultimate layer to memory. This certainly imposes limitations to the energy efficiency and memory footprint of the additional learning circuitry. However, such batch size is kept so to focus purely on the performance of RRAM-based on-line learning, trying to find what is the complexity level at which such approach breaks.
To analyse the effect of variability in the algorithm, cycle-to-cycle noise in the weight update is modelled with gaussian noise with varying standard deviation. Results are shown in Fig. 4.9b. The Baseline accuracy (horizontal dashed line) at 81.3% refers to the case without variability. The yellow bar highlights the range of variability of RRAMs and it is at this level that the accuracy starts decreasing considerably. As variability is correlated with the resistive level of RRAMs, these results hints to the fact the higher conductances

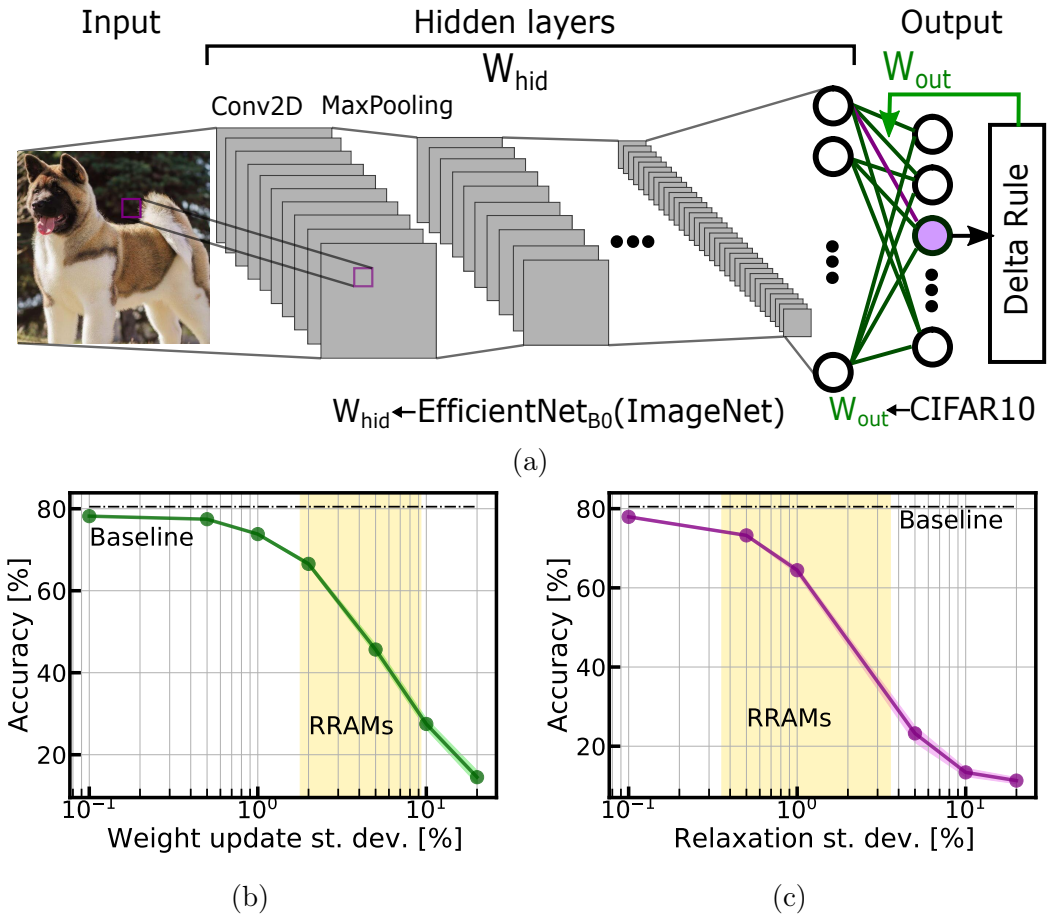


Figure 4.9: Variability analysis learning CIFAR10 online. (a) In this test, the weight update was performed adding random noise. Noise is injected from a Gaussian distribution with given standard deviation (x-axis). Noise is less tolerated than in the case of MNIST. The Delta Rule learning tolerates noise up 1% of standard deviation. The yellow bar represents the noise levels of RRAMs. (b) Cycle-to-Cycle variability effect on accuracy. C2C is modelled as a gaussian noise whose standard deviation is swept between 0.1% and 10%. RRAM noise magnitude is highlighted in yellow. The Baseline value of the Delta Rule with no variability is plotted with an horizintal dashed line. (c) Effect of RRAM's Relaxation on the performance.

are to be preferred. For example, 1T1R devices featuring large transistors allow for higher compliance currents, thus reaching higher conductive states.
Similarly, an analysis on the Relaxation effect is reported in Fig, 4.9c. Relaxation acts on all the devices, i.e. weights, producing a random noise on the conductance levels after a programming operation. Unlike cycle-to-cycle variability, Relaxation acts on all the devices and is not mitigated with the P_prog parameter, which acts by reducing the portion of parameters that are updated by the delta rule. The yellow bar highlights the relaxation levels in RRAMs. In the analysis, relaxation is modelled with gaussian noise

and results show that performance starts dropping with noise level at 1% of the maximal weight value.

Chapter 5

A step beyond in neuromorphic computing

This last chapter groups together two works that have one common aspect: to go beyond what is normally built in neuromorphic computing. As seen in the previous chapters, neuromorphic computing is based on the concept of neural networks where neurons are connected by synapses. In general, synaptic weights are assumed as the only learned parameters that adapt to a given task.
In the memristive-self-organized-network (MEMSORN) project, this paradigm is overcome and neurons are endowed with plasticity. An unsupervised learning scheme is developed where the plasticity of both neurons and synapses results in improved performance. MEMSORN is also technologically plausible and based on the utilization of RRAMs as plastic elements in both neuronal and synaptic circuits.
Biological neurons are much more than the point-like structure assumed by both deep learning and neuromorphic computing. They are complex structures with many parts and all contribute to computation. One such part is the dendrite, elongated branches which extend from the neuron's soma and connect to afferent synapses. Dendrites perform interesting non-linear and dynamical computations, which is probably fundamental for efficient information processing. Inspired by the role of dendrites, a dendritic circuit element was designed and built, forming dendritic networks to extend the functionality of neuromorphic systems.

5.1 MEMSORN: a memristive unsupervised self-organized map

Learning is a fundamental component for creating intelligent machines. Biological intelligence orchestrates synaptic and neuronal learning at multiple time-scales to *self-organize*

populations of neurons for solving complex tasks. Inspired by this, we design and experimentally demonstrate an adaptive hardware architecture *SOSN*. MEMSORN incorporates resistive memory (RRAM) in its synapses and neurons which configure their state based on Hebbian and Homeostatic plasticity respectively. For the first time, we derive these plasticity rules directly from the statistical measurements of our fabricated RRAM-based neurons and synapses. These "technologically plausible" learning rules exploit the intrinsic variability of the devices and improve the accuracy of the network on a sequence learning task by 30%. Finally, we compare the performance of SOSN to a fully-randomly-set-up recurrent network on the same task, showing that self-organization improves the accuracy by more than 15%. This work demonstrates the importance of the device-circuit-algorithm co-design approach for implementing brain-inspired computing hardware.

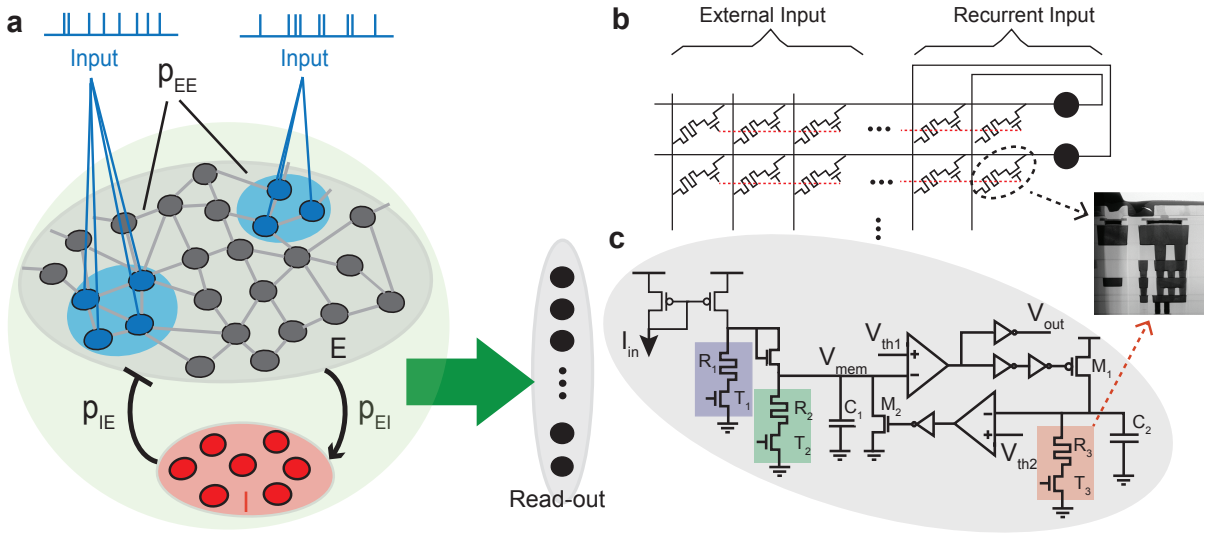


Figure 5.1: SRNN and its hardware implementation. (a) The SRNN is composed of a recurrent pool of excitatory neurons whose connections are formed by random fixed weights (static) or through learning (SOSN). The network is excited by spatio-temporal inputs activating sub-populations, shown in blue. Each of the sub-populations encodes a particular part of the sequence. The excitatory neurons are connected to an inhibitory population (shown in red), among which there are no recurrent connections. Both excitatory and inhibitory populations contribute to the activation of the output, via the readout connections (green arrow). Each neuron in the readout is assigned to a different prediction class. (b) A possible hardware implementation of the SRNN. Neuron's recurrent and external input connection are implemented by RRAM devices assembled in a crossbar array. Rows of the crossbar are connected to the inputs, while the neurons are connected to its columns. (c) Neurons are implemented using a hybrid CMOS/RRAM design. RRAM hold the parameters of the neurons, such as gain (purple 1T-1R), leak (green 1T-1R) and refractory period (red 1T-1R).

5.1.1 Motivation

The hallmark of intelligence is the ability of the brain to adapt and *self-organize* itself to sensory information it receives throughout its lifespan. This self-organization is mediated by a rich set of neuro-cognitive mechanisms that together contribute to sequence learning and long-term memory formation [262]. While learning, a web of memory forms between large groups of neurons, leading to coherent dynamic activity patterns that are a function of the sensory information the system receives.

It has been shown that the combination of brain-inspired learning rules at different time-scales lend themselves to the self-organization of dynamic networks for behavior control [263, 264]. This type of self-organization lies in the unsupervised learning realm where the ground truth is not available for learning. Instead, the memory forms as a result of clustering information in *cell-assemblies* [263]. A cell-assembly can be defined as a group of neurons with strong mutual excitatory connections. Once a subset of a cell-assembly is stimulated, its neurons tend to be activated as a whole, so that the cell can be considered as an operational unit of a SRNN. Applying local learning rules to the recurrent connections forms independent cell-assemblies and makes the SRNN more powerful in extracting temporal features in the data, compared to a fully-randomly-connected solution [264]. One example of such a concept has been shown in SORN [264], a recurrent network model of excitatory and inhibitory binary neurons. It incorporates a Hebbian-based synaptic plasticity at a shorter timescale, along with Homeostatic plasticity at a longer timescale. It is illustrated that SORN outperforms random RNN without plasticity on sequence prediction tasks.

Implementing SORN-like networks on a hardware substrate holds great promise for machine intelligence and autonomous agents, especially in situations where the agent is in unknown environments [183, 265]. Neuromorphic technologies with online learning capabilities can support the hardware implementation of such self-organizing SRNN [25, 266].

On-line learning in electronic devices requires local and distributed memory elements for storing the learned parameters (e.g., the synaptic weights). RRAM has recently gained significant attention as a promising memory technology for on-line learning [131, 267, 132, 133, 130, 268, 269, 270, 271]. Its non-volatile and multi-state properties makes it a plausible candidate for employment as adaptive hardware. Importantly, its internal dynamics and intrinsic stochasticity have been proven beneficial for on-chip learning [5, 229, 272, 273] which cannot be simply introduced in a digital implementation [274, 25]. As biological networks rely on small unreliable components for reliable learning, they can provide guidance for learning with RRAM devices. Brain-inspired unsupervised Hebbian learning strategies have already been extensively explored in adaptive memristive neuromorphic architectures [275, 276, 130, 277]. In these works, the RRAM conductance changes towards a more/less conductive state based on the correlation/anti-correlation between its pre- and post-synaptic neurons. However, Hebbian learning by itself cannot robustly lead

to self-organization, as it implements a greedy mechanism which can lead to unstable dynamics [278]. To achieve self-organization in memristive neuromorphic architectures, a multitude of plasticity mechanisms need to be at play together, with properties and dynamics that match the physics of the underlying adaptive hardware substrate [124, 153].

Here we present *SOSN*: a hardware architecture inspired by SORN with multi-timescale on-chip plasticity rules. MEMSORN is developed following a device-algorithm co-design approach exploiting the physics of the employed RRAM devices taking advantage of their variability. We designed and fabricated the RRAM-based synapse and neurons in 130 nm CMOS technology integrated with HfO_2-based RRAM devices. Based on the statistical measurements from these designs, we derive the local *technologically plausible* plasticity mechanisms (Hebbian and Homeostatic), and apply them in the SOSN architecture. We benchmark the network on a sequence learning task, and show that this approach exploits the intrinsic variability of the RRAM devices and improves the network's accuracy as a function of sequence length, learning rate, and training epochs. As a control experiment, we apply the same task to the same exact network, only without learning, whose recurrent connections are randomly set up. We show that SOSN accuracy outperforms the random network by about 15%. This work represents a fundamental step toward the design of future *neuromorphic intelligence* devices and applications.

Results Inspired by SORN [264], we implemented two recurrently-connected networks of LIF neurons: one randomly connected with fixed weights (static) and one with connections that change through learning (SOSN). Other than this difference, the two networks are identical. Both networks consist of an excitatory pool of recurrently connected neurons, and an inhibitory pool of neurons that inhibit the excitatory ones, along with a read-out layer fully connected to the two pools. The inhibitory neurons balance the activity of excitatory neurons by providing a negative feedback [279, 280]. Inspired by neuro-anatomy considerations on cortical circuits, we divided the excitatory and inhibitory population to 80% and 20% of the total number of neurons, respectively [281]. Different sub-populations of neurons are stimulated by different parts of the input sequence. In both networks, the activities of all the recurrent neurons is fed to a linear classifier at the readout which learns to distinguish between different classes of input (see Fig. 5.1a).

5.1.2 Endowing both Synapses and Neurons with plasticity: hardware implementation

To implement the network in hardware, we designed a crossbar memory architecture (Fig. 5.1b). Its rows are connected to the neurons and its columns are connected to either external inputs or to a recurrent input from another neuron. We employed RRAM both in the design of the synapses at the cross-points holding their strength (Fig. 5.1b), and in the design of the neurons holding their internal parameters (Fig. 5.1c); Each synapse contains a transistor in series with an RRAM (aka 1T-1R), with the free side of the transistor and

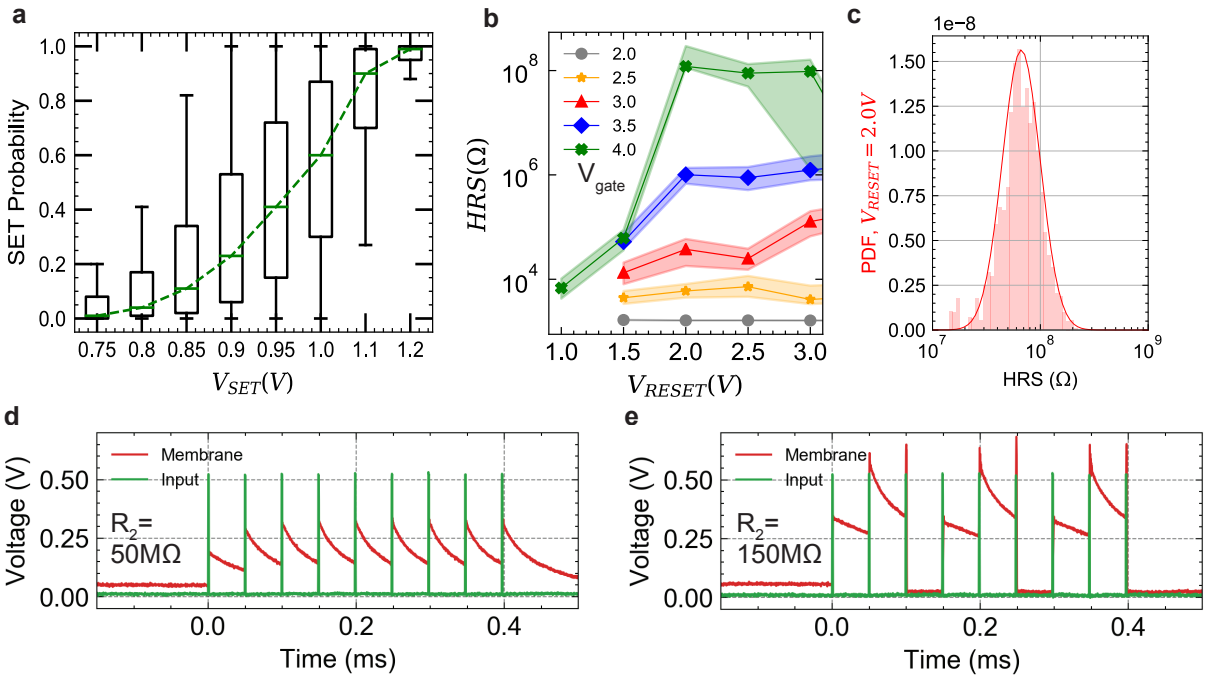


Figure 5.2: Measurements from our fabricated RRAM array and synapse and neurons in 130 nm technology. a,b,c) Experimental results from the fabricated 4 kb synapse array, each device is programmed 100 times. a) SET characteristics. The box plot represents the SET probability as a function of the SET voltage, the green horizontal bar represents the median value, the box lower and upper limits represent the ±25% and ±75% quantile respectively, and the whiskers show the ±95% quantile. (b) RESET characteristics; HRS measurements as a function of the RESET voltage applied across the devices, for different gate voltages applied to the transistor (T). (c) The HRS distribution at $V_{RESET} = 2.0V$ and $V_{gate} = 4V$ which fits well to a log-normal distribution. (d), (e) Experimental results from the fabricated neuron. The neuron is excited by a train of spikes with a pulse width of $1\mu s$ and a magnitude of $450\,mV$ and a frequency of $1\,kHz$ (green). Changing the state of R_1 and R_2 devices changes the gain and time constant of the neuron, which does not fire in d) due to a weak R_2 value, but fires multiple spikes in e) thanks to a larger R_2 resistance.

the RRAM connecting to the rows and the columns, respectively; Each neuron implements the LIF model shown in Fig. 5.1c. This hybrid CMOS/RRAM neuron design encompasses three RRAM whose value set the neuron time constant (shown in green), gain (shown in purple) and refractory period (shown in red) [154]. The adaptive nature of RRAM allows for learning both the synaptic and internal neuron parameters in an on-chip and online fashion.

As soon as an input spike arrives to a column, a voltage is applied across the corresponding synaptic RRAMs, giving rise to a current, through Ohm's law. All currents are summed at the rows, and are integrated by the corresponding neurons [229]. The input to the

neuron is multiplied by the gain ($R1/R2$), and is integrated on the membrane capacitance C_1 with a time constant determined by R_2C_1. As soon as the voltage on C_1 passes the threshold V_{th1}, the neuron generates a voltage spike, and sends it both to V_{out} and to the feedback path. In the feedback path, the neuron's spike is integrated on C_2, and the resulting voltage has a time constant of R_3C_2. As soon as this voltage passes the threshold V_{th2}, the membrane capacitance C_2 is reset, and the neuron awaits the next input current.

Synapse and neuron characteristics We fabricated and measured a 4 kb synaptic crossbar array along with the hybrid CMOS/RRAM neurons, using 130 nm CMOS technology integrated with HfO_2-based RRAMm in a 200 mm production line. The Front End of the Line, up to metal 4, has been realized by ST-Microelectronics, while from Metal 5 upwards, including the deposition of the composites for RRAM devices, the process has been completed by CEA Leti. RRAM devices are composed of a 5 nm thick HfO_2 layer sandwiched by two 5 nm thick TiN electrodes, forming an $TiN/HfO_2/Ti/TiN$ stack. Each device is accessed by a transistor composing the 1T-1R unit cell. The size of the access transistor is $650\,nm$ in width. 1T-1R cells are integrated with CMOS-based circuits by stacking the RRAM cells on the higher metal layers.

In the synapses, we can induce a change by applying a voltage across the RRAM devices. The device state changes from a HRS to a HRS (SET operation) by applying a positive voltage between the positive and negative terminals of the 1T-1R, while applying a voltage to the gate of the transistor, V_{gate}, to control the current passing through it during programming. Alternatively, the device switches from HRS to HRS, by applying a negative voltage across the 1T-1R (RESET operation). Both SET and RESET operations produce changes in a stochastic manner. This results in a distribution over the resistance values given a programming condition [282, 283, 284]. Typical values for the SET operation are Vgate in [0.9 - 1.3] V , while the Vtop peak voltage is normally at 2.0 V . For the RESET operation, the gate voltage is instead in the [2.75, 3.25]V range, while the bottom electrode is reaching a peak at 3.0 V . The reading operation is performed by limiting the Vtop voltage to 0.3 V , a value that avoids read disturbances, while opening the gate voltage at 4.5V. We define a threshold at $50\,k\Omega$ for the resistance marking the border between HRS and HRS, and characterize the SET and RESET properties; Fig. 5.2a shows the probability of the SET operation as a function of the voltage applied to the 1T-1R cell, which follows a sigmoidal function. The RESET operation is characterized in Fig. 5.2b as a function of the voltage applied across the devices, with different gate voltages. The distribution of HRS values for a RESET voltages of $2\,V$ is shown in Fig. 5.2c. The distribution fits well with a log-normal function [284].

In the neurons, we measured the output firing pattern in response to a spike train as is shown in Fig. 5.2 d, e and f. Setting R_1 and R_2 with different values increases (Fig. 5.2 d) or decreases (Fig. 5.2 e, f) the neuron's time constant, and thus changes the likelihood of the neuron firing. In sensory-motor applications, matching the dynamics of sensory signals to those of the electronic circuit in the processing hardware can minimize the system power

consumption and maximize the SNR [183]. Therefore, to obtain neuron's time constants of millisecond range, on the order of sensory signals, while limiting the size of the capacitors (to minimize area usage), the neuron's RRAM devices should be operated in their HRS ranging from MΩ to GΩ (Fig. 5.1b).

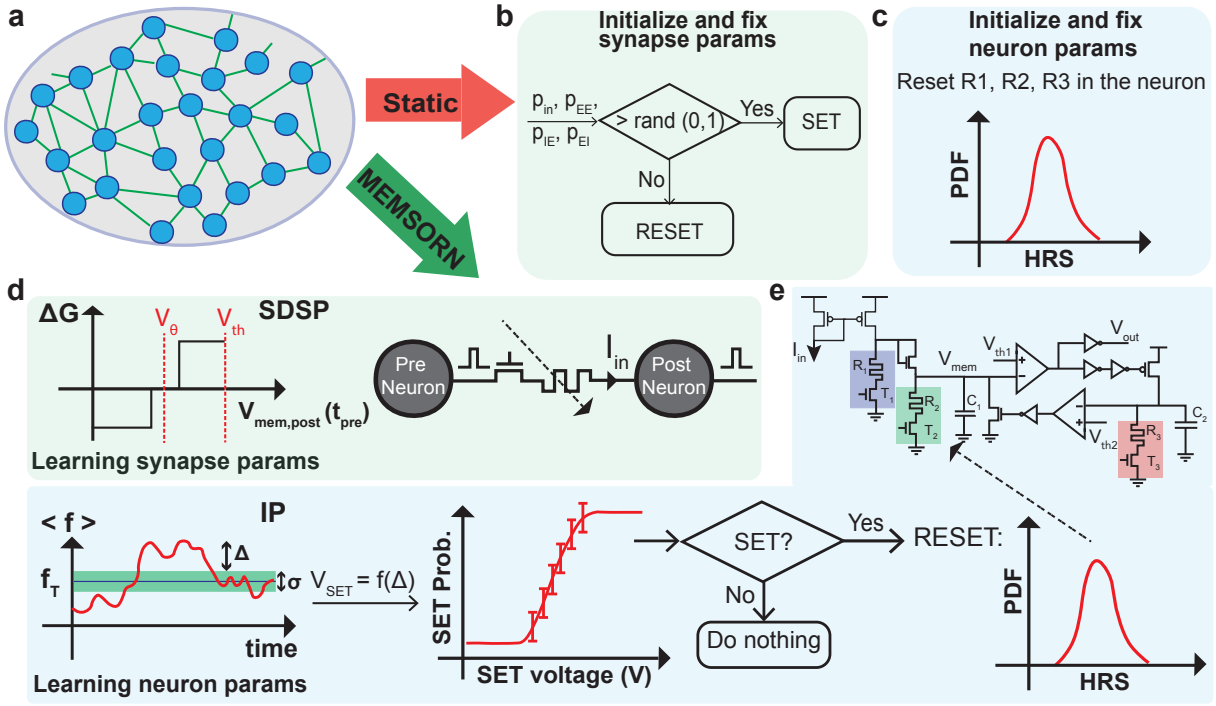


Figure 5.3: Technologically plausible algorithms for static and SOSN networks. (a) An SRNN with excitatory connections. (b, c) Synapse and neuron parameters for static network. Both values are fixed after an initialization process. (b) Synaptic parameters are initialized based on comparing the probability of different connections with a random number. (c) Neuronal parameters are initialized by resetting the resistors R_1, R_2 and R_3 which is equivalent to sampling from a log-normal distribution around a mean resistance that is a function of the reset voltage. (d, e) Synaptic and neuron parameters for SOSN network. Both parameters are learned throughout the input sequence presentation. (d) Synaptic parameters are learned based on the Hebbian-based SDSP learning rule. At the time of the pre-synaptic event (t_{pre}), weight (conductance) of the synapses are increased/decreased, if the membrane potential of the post-synaptic neuron ($V_{mem,post}$) is higher/lower than V_θ. (e) Neuron parameters are changed based on the IP algorithm which tries to keep the firing rate of each neuron ($< f >$) in a healthy regime ($f_T \pm \sigma/2$). If the neuron's firing rate goes beyond this regime, neurons' R_2 is first SET probabilistically and then RESET. The RESET process samples a new value for R_2 from the log-normal distribution of HRS values.

5.1.3 Technologically-plausible unsupervised learning

With the *technologically plausible* algorithm design, we aim to optimize the hardware implementation of algorithms by taking the hardware physics into account while developing the algorithm. Figure 5.3 depicts the algorithms for the two static and SOSN networks which are derived based on the synapse and neuron measurements of Fig. 5.2.

Static network The algorithm for static network (i.e., with fixed random weights) is depicted in Fig. 5.3b, c. The synapse and neuron behavior is fixed *a priori*; the synaptic connections are set randomly, by comparing the probability of connections in different populations to a random number, and if higher/lower, induce a SET/RESET to the devices (Fig. 5.3b); The neuron parameters are sampled from the HRS log-normal PDF derived from the measurements of Fig. 5.2c, equivalent to applying $V_{RESET} = 2\,V$ to the devices.

SOSN The *SOSN* plastic network *self-organizes* to form multiple cell-assemblies. This is done through changing the RRAM in the synapse and neuron parameters through learning. The excitatory synapses undergo a Hebbian-type plasticity, i.e., SDSP, which changes the synaptic RRAM based on the correlation between the input (pre-synaptic) and output (post-synaptic) neural activities [285, 266]. In addition, the neuron parameters undergo Homeostatic plasticity, i.e., IP, which acts as a regulatory mechanism to keeps the neuron's firing activity within a desired range [4]. Both forms of plasticity are well suited for the implementation on CMOS and RRAM hardware.

Following the SDSP rule, the RRAM resistance of a synapse is decreased/increased, on the onset of its pre-synaptic spike, if the membrane potential of the post-synaptic neuron is higher/lower than V_θ threshold (Fig. 5.3d). The measure of correlation in SDSP is the difference between the membrane potential of the post-synaptic neuron V_{mem} to a defined threshold, V_θ at the time of the pre-synaptic spike t_{pre}. The weight update on t_{pre} is defined as:

$$w_{EE} = \begin{cases} w_{EE} + LR, & \text{if } V_{mem} \geq V_\theta \\ w_{EE} - LR, & \text{otherwise} \end{cases}$$

Where w_{EE} is the weight between the excitatory neurons, and LR is the learning rate. The SDSP rule is thus controlled by two parameters, the thresholds applied to the post-synaptic neuron membrane voltage (V_θ), and the synaptic weight increment (LR).

On the other hand, IP changes the neuron's RRAM to maintain its output firing rate, f_n, close to a target firing frequency, f_T, within a tolerance of σ (Fig. 5.3e). If f_n lies outside of these boundaries, the RRAMs in their HRS are updated accordingly.

A target firing frequency f_T with an error margin σ is defined as the desired range, and the neuron measures its firing rate f_n with respect to the boundaries $f_T \pm \sigma/2$. If f_n moves outside of these boundaries, the value of HRS needs to be updated. To do so, the RRAM

Algorithm 4: IP algorithm

```
Initialization: R = RESET_init(V_Reset)
while t < taskDuration do
    for Neurons in the excitatory pool do
        if @t_post: |f_neuron − f_T| > σ/2 then
            # Sub-threshold Stochastic SET
            V_set = f(|f_neuron − f_T|)
            p_set = P_subthSET(V_Set)
            if R_final < 50 kΩ then
                # RESET
                R_HRS = RESET()
```

is SET with a subtheshold SET voltage which is proportional to the difference between the target and neuron activity. The higher the difference, the higher the SET voltage and thus the higher the probability of setting the device. If the device is SET (i.e., the final resistance $R_{final} < 50\,k\Omega$), we then RESET the device to sample from its internal distribution and find a new value that sets the time constant and gain of the neuron.

For simplicity, we have chosen to only update R_2 which simultaneously changes both the gain and the time constant of the neuron. Changing the gain will additionally implement synaptic scaling which is another homeostatic plasticity mechanism, used in conjunction with IP in the original SORN paper [264]. To tune R_2 in HRS, it is first SET and then RESET. SET is done probabilistically proportional to the difference between f_n and f_T (δ). Once SET, The RESET operation with a fixed V_{RESET} effectively samples a new HRS value from a log-normal PDF. Therefore, neurons with a frequency deviating significantly from the target will change their leak and gain by acting on R_2, to adapt their firing rate. Note that since the amplitude of V_{RESET} is fixed, the sampled HRS value is drawn from a single distribution, which makes the search for the correct resistance values *non-guided*. This reduces circuitry overhead with respect to an alternative algorithm in which the RESET operation is performed by adapting the V_{RESET} to the deviation of the f_n from f_T (i.e. $V_{RESET} \propto |f_n - f_T|$) [4].

Benchmark

To validate our approach, we used the same benchmark proposed in the original SORN paper [264]: a sequence learning task based on counting for predicting the next sequence at the output. The network receives a shuffled alternation of two input sequences of length $n+2$ of 6 possible characters in $[A, B, C, D, E, F]$. In both sequences, either characters, B or E are repeated n times (Fig. 5.4a). Examples of these sequences are $S_{1,n} : [A, B_1, B_2, ..., B_n, C]$ and $S_{2,n} : [D, E_1, E_2, ..., E_n, F]$. The goal is to learn to predict the next character given all the previously-presented ones, *i.e* $P(next - character_i | \sum_j^{i-1} shown - character_j)$. After

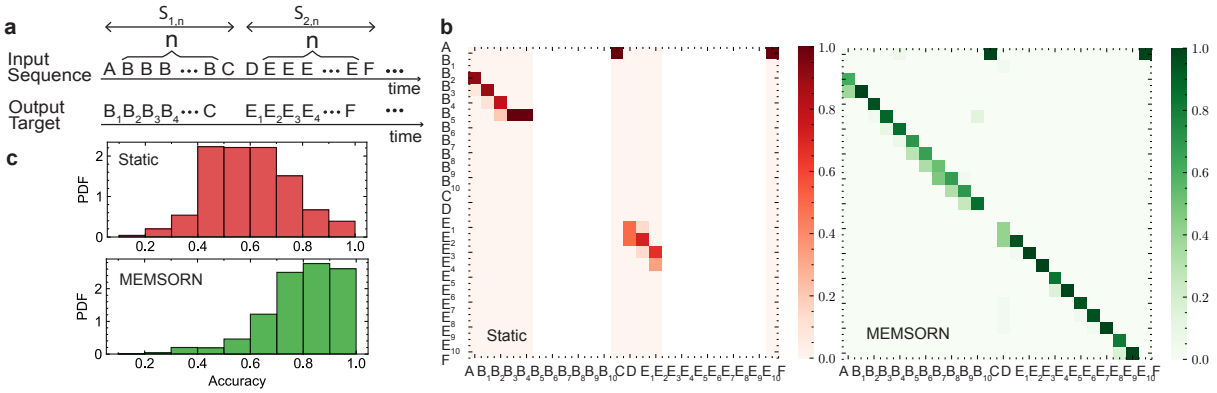


Figure 5.4: Static and SOSN performance comparison. (a) Sequence learning task. Two input sequences of $S_{1n} = ABB...BC$ and $S_{2n} = DEE...EF$, with Bs and Es presented n times, are fed to the network. Each letter represents part of the sequence. The task is to predict the next symbol in the sequence, for which it needs to keep the count of the number of presented B and E symbols. (b) Confusion matrix for the static (red) and MEMSORN (green) networks. The static network is capable of separating only the first part of the sequence, whereas the SOSN network can successfully predict the next letter in the sequence resulting in a diagonal confusion matrix. Understandably, performance drops to chance level between the two sequences, since input symbols are equiprobable and their temporal succession carries no structure. (c) Histogram of the accuracy in the static (red) and SOSN (green) networks tested on 1000 different networks, for the counting task with sequence length (n) of 10. The mean of the accuracy distribution in SOSN network increases compared to the static network (mean of 0.756 for SOSN network compared to 0.596 in the static network). Also, the number of low-accuracy networks in SOSN is greatly reduced compared to the static network (about four times).

fixing the length of the sequence, the network has to learn to count the repetition of characters B and E by means of a reliable dynamical state.

We applied the counting task to the static and SOSN networks, initialized with the parameters shown in 5.1, and compared their performance. The network is asked to differentiate between $n = 10$ repetitions of the same symbol, presented in the middle of the two sequences. Each symbol's position in the two sequence is assigned to one output neuron in the readout whose activity represents the network's prediction of the next symbol. Fig. 5.4b shows the confusion matrix, indicating the predicted, compared to the expected output.

The static network is capable of separating only the first repetitions, whereas the SOSN network can successfully resolve all the repetitions, forming a diagonal in the confusion matrix, matching the output to the target. Since the two sequences are randomly alternated, the output of the network under the presentation of the last symbol in the sequence cannot be predicted. This applies to outputs 0, 13 and 25 in Fig. 5.4b. The internal dynamics in the static network saturates and lands on an attractor state from which no further

Neurons			Synapses		SRNN	
	Excitatory	Inhibitory				
num. neurons	160	40	τ	1 ms	p_{EE}	2%
R_2	(IP)	1 $G\Omega$	Weight	(SDSP)	p_{II}	0%
R_1	400 $M\Omega$	600 $M\Omega$			p_{EI}	2%
C_1	10 pF	10 pF			p_{IE}	10%
τ_{Ca}	100 ms	100 ms				
V_{th}	0.2 V	0.2 V				
R_3	1 $G\Omega$	1 $G\Omega$				
C_2	2 pF	2 pF				
IP			SDSP			
F_T	50Hz		V_{th}	0.2 V		
σ	15Hz		V_θ	0.1 V		
V_{RESET}	2 V		$LearningRate$	0.01-0.1		

Table 5.1: Parameter values for the initialization of the SRNN. Such values are defined with small-to-absent tuning, with the only aim to guarantee a minimal activation of the network, so to fully rely on the plasticity mechanisms (SDSP and IP) to improve performance. Some parameters, such as the magnitude of RRAM resistance in HRS and the Membrane Capacitance, are forced by technological constraints.

information can be extracted. The SOSN network, instead, is capable of forming more complex dynamics that allow for fading memory to form and separate the repetitions in the input sequence. Figure 5.4c illustrates the histogram of the accuracy calculated over 1000 networks initialized differently for both networks on the counting task with the sequence length of $n = 10$. As shown, the mean accuracy of the SOSN network is increased compared to the static network. (Mean accuracy of 0.756 compared to 0.596 respectively). The standard deviation is due to the random initialization of the connections and the variability of RRAM, implementing both the weights of the connections and the parameters of the neurons Taking into account the hardware constraints, our statistical analysis shows that by enabling learning inside the recurrent network, there is a higher probability of obtaining a more accurate network; i.e. the number of learned networks that can correctly predict the next letter with an accuracy of more than 0.8, is four times that of the static network.

5.1.4 Analysis on the effect of variability in SOSN

RRAM devices undergo cycle-to-cycle and device-to-device variability as was confirmed with our measurements in Fig. 5.2. To understand the effect of variability in SOSN, we performed simulations on four cases: (i) No device variability and IP operation off; (ii) Variability in devices receiving the SDSP rule, and IP operation off; (iii) Variability in devices receiving the SDSP, and IP operation on without variability; (iv) Variability in both

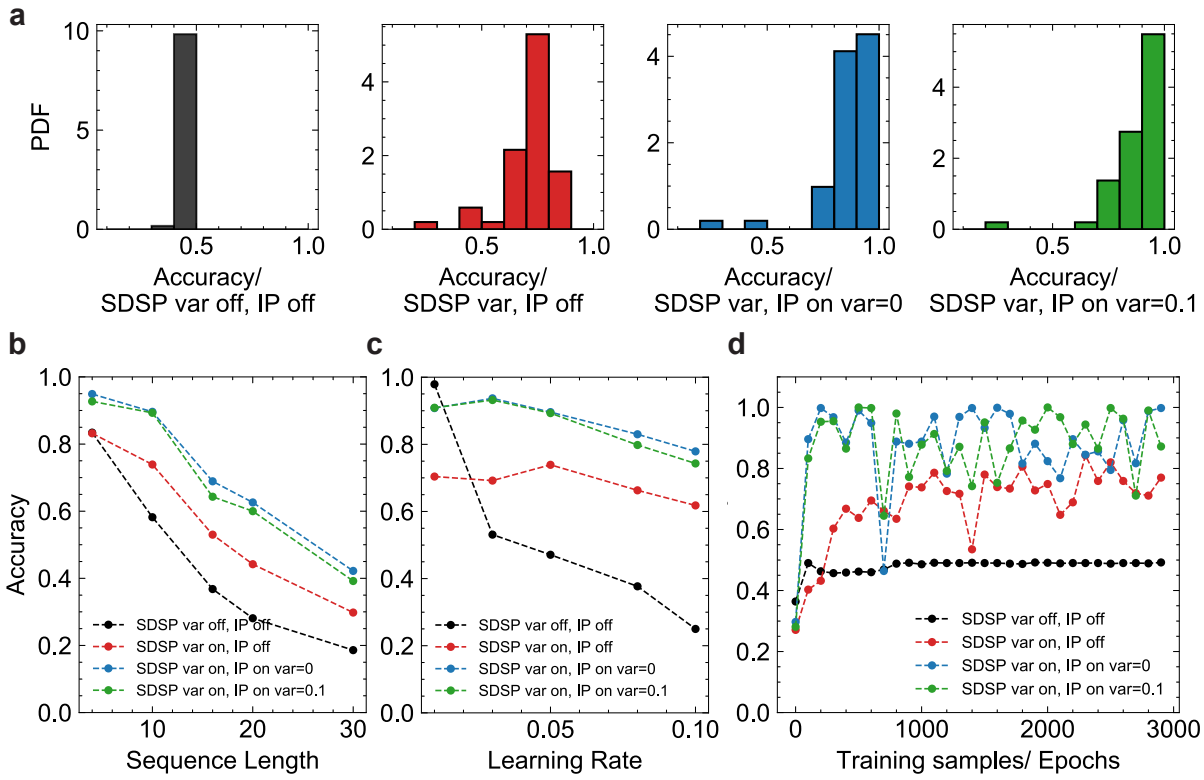


Figure 5.5: Performance of our proposed self-organized network (SOSN) under four different cases of variability in the device models: (i) No device variability and IP operation off (black), (ii) Variability in devices receiving the SDSP, and IP operation off (red), (iii) Variability in devices receiving the SDSP, and IP operation on without variability (Blue) and (iv) Variability in both SDSP and IP learning with standard deviation for the IP devices set to 0.1, taken from our measurements (green). (a) Histogram of accuracy for 500 networks confirms the higher accuracy for the networks including variability and IP operation compared to other conditions. (b) Accuracy of the SOSN network on the counting task with respect to the sequence length. As the sequence length increases, the task becomes harder: introducing variability, calibrated on measured data, helps the accuracy of the network as all the cases with variability outperform the case without any variability. (c) Average accuracy of the SOSN network on the counting task with length of 10 as a function of different learning rates. Introducing variability makes the network robust to hyper-parameter change. (d) Learning evolution of the network accuracy on the counting task with sequence length of 10 for the four variability cases. Condition (i) has much less noise, but has an overall lower accuracy (less than 40%) than the cases where variability and IP are introduced.

SDSP and IP learning with standard deviation for the IP devices set as 0.1, taken from our measurements. It is worth noting that in condition (iii), since there is no variability in IP operation, the same initial value is always applied when IP is acting.

Figure 5.5 shows the network performance under these four conditions. The figure demonstrates the positive effect of the regularizing IP mechanism, and how SOSN network exploits the different sources of variability of the RRAM devices to increase its accuracy on the sequence learning task; Figure 5.5a plots the histograms of accuracy for every 100 samples of learning in the SOSN network for all the variability conditions. The histograms show that introducing IP operation, and any source of variability shifts the mean of the accuracy of the network to higher values; Figure 5.5b illustrates the accuracy as a function of the sequence length. As the sequence length increases, the network needs to remember increasingly longer sequences which tests its fading memory [286]. Thus, the accuracy of the network drops with longer sequences. It is worth noting that as the sequence length increases, the number of output neurons increases, and thus the baseline chance level accuracy reduces. Figure 5.5b confirms that the networks including IP and added source of variability outperform other conditions. Figure 5.5c depicts the network accuracy as a function of the SDSP learning rate. Despite that large learning rate results in a consistent drop in accuracy, introducing variability suppresses accuracy degradation. This suggests that the noise introduced by the variability of the RRAM devices is beneficial for the stability of the network making it less sensitive to hyper-parameters and low bit resolution. This is because through learning with noise, the algorithm finds a set of parameters that are more insensitive to noise. Figure 5.5d shows the accuracy evolution of the SOSN network during learning epochs. Each epoch consists of presenting one of the two sequences which are presented to the network with a random order. Condition (i) without any variability and IP operation (black) leads to a more stable learning dynamics, but also lower performance. Instead, adding noise to SDSP or adding the IP operation causes some instability in the network, but also allows for much higher overall accuracy. Finally, combining the variability in SDSP with that of IP leads to the best performance compared to other conditions.

The positive effect of variability is because a distribution of parameters due to variability provides a larger space of parameters for search during learning which helps the network to explore and reach a better set of parameters for the task.

Clustering analysis

To understand the dynamics of the static and SOSN networks, we performed clustering analysis on the firing activities of the neurons inside the excitatory pool. Figure 5.6 shows the result of the clustering analysis. First, we reduced the dimensionality of the neural activity using PCA. Figure 5.6a plots the PCA of both network activities in response to 50 sequences of length 10. Temporally adjacent letters in the sequence line up next to each other in the principle component space. This indicates the higher structural richness in SOSN compared to the static network. Moreover, this helps with the classification accuracy in the readout layer, since the sequences become more linearly separable as indicated by the PCA plot. Figure 5.6b plots the histogram of explained variance in the firing rate of the random and SOSN networks with respect to the first 20 principle

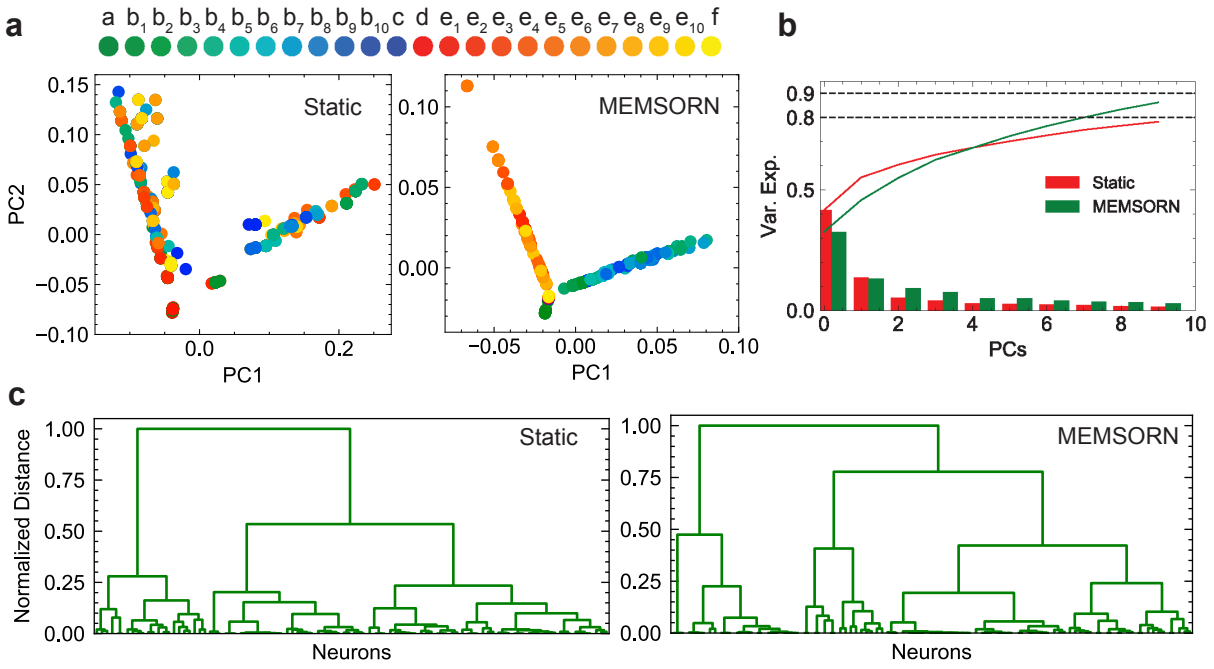


Figure 5.6: Clustering analysis on the spiking activity of the networks for static and SOSN architectures. (a) Principle Component Analysis (PCA) applied to the firing rate of the two networks in response to 50 sequences of length 10 (600 letters). Each color is assigned to a different position of the letter in the sequence with similar colors encoding temporal adjacency of the letters in the sequence. (b) Histogram of captured variance by the first 20 PCs in the static and SOSN networks. The explained variance amounts to 79% for static network compared to 87% in the SOSN one suggesting more orderly dynamics in SOSN network. (c) Dendogram of static and SOSN networks showing the hierarchical relationship between clusters of neurons. The normalized height of the dendogram indicates the distance between the clusters and the links indicate the order in which the clusters are joined. For any given distance, the number of branch numbers for MEMSORN are larger than those for the static network, indicating that the clusters in MEMSORN are better-structured.

components. The explained variance is about 11% more in SOSN network compared to the random network suggesting more orderly dynamics in SOSN network. Additionally, we performed a hierarchical clustering analysis on the activity of the SRNN which reveals the formed cell-assemblies. The result is indicated by the dendogram in Fig. 5.6c, showing an increase in the number of uncorrelated clusters in SOSN network compared to the random network. This is as a result of more structure emerging from the learning in the recurrent network which is in agreement with the unsupervised memory formation in cell-assemblies as we argued in the introduction.

5.1.5 Energy and latency estimations

The MEMSORN hardware reaches convergence for the sequence learning task within the presentation of 1000 learning samples (Fig. 5.5d), giving rise to ≈ 13 minutes for learning the task. The power consumption of the MEMSORN's recurrent network during this task consists of three components of static power consumption, dynamic power as a result of the firing of the neurons, and dynamic power due to changing the state of the RRAM devices for learning. The static power consumption of the recurrent layer, including the 1T-1R array and the 200 neurons is 0.2 μW. The dynamic power due to the firing of the neurons is about 0.8 μW, and the dynamic power of changing the state of the RRAMs due to IP and SDSP learning is 0.2 μW. For the duration of the learning, this gives rise to 936 μJ of energy consumption. These values are well within the energy budget and real-time online learning requirement of edge devices [287].

Latency and energy calculations

As is shown in Fig. 5.5d, the learning takes about 1000 presentation of samples to converge. Each symbol is presented for 50 ms with a 200 ms of wait time in between pattern presentations. For a pattern of 12 symbols (n=10), that gives rise to 800 ms per epoch. Therefore, the total latency is 800 seconds, or 13 minutes. We identified three sources of power consumption in our system: static power, dynamic power due to the firing of the neurons, and dynamic power due to the state of the RRAM devices changing.

Static power: The static power consumption of each neuron, together with the switches is 1.4 nW. This gives rise to ≈ 0.2 μW of static power consumption for the entire population of the neurons.

Neuron dynamic power Based on our measurements, the energy consumption of our neuron in 130 nm process is 100 pJ/spike. Our recurrent layer has 160 excitatory neurons firing at 50 Hz (maintained by the IP algorithm). This gives rise to: $160 \times 50\ spikes/second \times 100\,pJ/spike \approx 0.8\mu W$.

RRAM state change dynamic power During the learning operation, the state of the RRAM devices changes due to the IP and SDSP learning rules. We have counted the total number of times that the state of the devices are changing during the learning process, which is $\approx 3 \times 10^6$ times. As we have previously reported [154], each RRAM SET and RESET cycle consumes around 50 pJ. For the duration of the learning (13 minutes)x, this gives rise to $50\,pJ \times 3 \times 10^6/(13 \times 60\,seconds) = 0.2\,\mu W$. Therefore, for the duration of the learning process, the total energy consumption is $(0.2\mu W + 0.8\mu W + 0.2\mu W) \times 13 \times 60\,seconds = 1.2\,\mu W \times 780\,seconds = 936\,\mu J$.

5.2 Dendritic Computation

Neuromorphic computing develops around the idea of bio-inspired computation, following the brain's primitive computational elements. At this stage of its evolution, neuromorphic computing mainly settled for systems composed of neurons and synapses. It is a compromise found by minimizing the system complexity and still obtaining good computational power. But to develop more advanced neuromorphic solutions, a question arises: **is there more to computation in the brain than neurons and synapses?**
The answer is of course yes, as it is well known in neuroscience that neurons are not point-like structures, but rather complex elements with many parts. So-called "compartments models" [288] of neurons describe most of - but not all - the different components of neurons, from the dendrites, to the soma and to the axons. These models greatly increment the complexity of neurons and are not easily simulated by conventional computers, reason why their usage is limited to small micro-circuit in neuroscience. The purpose of complex compartment models is explaining or understanding particular dynamics observed in the brain, rather than it is building efficient computers. However, it is intuitive to think that such complexity - or some part of it - is what allows a biological neuron (and a brain) to be so computationally powerful and efficient. More efforts have to be made in order to understand where the added complexity introduced by these highly non-linear and dynamical components of biological neurons pays off in pragmatical terms for computation. What are the features that can be borrowed from biological neurons that allow neuromorphic computing the next step forward?
A lesson can be learned from Deep Learning: the evolution of Recurrent Neural Network. It is well-known that vanilla RNNs fail at solving complex task, especially in cases where one deals with long sequences. It is because of the vanishing or exploding gradient problems. However, RNNs are among the most used architecture for Deep Learning at the edge, for speech recognition and temporal signals processing. How have RNNs become so powerful and ubiquitous? It is because of specific added complexity to the architecture that greatly improve the potential of the network. LSTMs [289] and GRUs [290] allow to mitigate the vanishing gradient problem of vanilla RNNs with an ad-hoc construction of feedback connections and "forget" gates allowing gradients to be propagated easily further back in time. These mechanisms proved robust to the back-propagation-through-time algorithm required to train RNNs and induce long term memory. Can Dendrites be for SNNs what "forget" gates are for RNNs?

Dendrites in biology Dendrites are protoplasmic protrusion extending from the body of a neuron, receiving electro-chemical stimuli from neighbouring neurons and propagate them to the neuron's soma. Action potentials - the neuronal communication signals called spikes - are generally produced in the soma, transmitted to axons and received by dendrites, to then be integrated by the receiving neurons. The connection between axons and dendrites are the synapses, forming a cleft between two lipidic layers, through

which neurotransmitter and ions travel and communicate spikes across neuronal circuits. Dendrites provide a large surface on which synapses are formed and adapted over time. The size and number of dendrites per neuron depends on the neuron's type, which can be either multipolar, bipolar or unipolar. Multipolar neurons are commonly found in the cortex and possess multiple dendrites and one large axonal branch; bipolar neurons feature one dendrite and one axons; unipolar neuron are mainly used to perceive sensory signals from a dendritic termination. Dendrites can arborize and form an impressive number of dendritic spines, post-synaptic connections from afferent axons. Dendritic spines can be as many as 15000 per neuron and a dendritic branch can receive as many as 100,000 inputs from neighbouring neurons [291, 292]. As much as synapses, dendrites also are plastic elements: their shape, size and thickness are thought to play a great role in memory formation [293, 294, 295]. Dendrites tend to modify their characteristics and morphology more in the early years of a mammal's life than in the adult life. It has been observed that degradation of dendrites is a common factor in different neuropsychiatric disorders [296]. However, as neuromorphic engineering, dendrites are mostly interesting for their unique electrical properties. The most basic of which is the transport of electrical signals from synaptic ions release. Action potential are propagated through the length of the dendritic branch as a signal through a cable. Indeed, cable theory is often utilized to describe the propagation of the action potential from the dendritic spines through the soma [297]. Such an action results in two main effects on the action potential: a delayed introduced by the propagation and a modulation due to the lossy transmission on the potential. The length, width and shape of the dendrites influence the property of propagation of the action potential to the soma. Furthermore, dendrites also feature ion-gated channels that participate to complex dynamical processing of the incoming stimuli. It results that Dendrites exhibit a wide range of behaviors useful for computation. Among the most important: coincidence detection, introduction of transmission delays, non-linear summation of input signals.

5.2.1 The Dendritic circuit element and the Dendritic Network

We take inspiration from such interesting features of dendrite to build a circuit that introduces delays in propagating spikes between neuron as well as modulating the amplitude of the transmitted signal. Of the many processing features dendrites are endowed with, we focus on the ability to introduce a delay and to modulate the transmitted spikes. Biological Dendrites (Fig. 5.7a) receive inputs from afferent axons and those are weighted by synapses. Then, action potentials are transmitted through the dendritic branch with given delay, before being integrated by the soma. To endow neuromorphic system with such computational abilities, we designed the circuit in (Fig. 5.7b). The Dendritic Element, as it is called, is a circuit that divides into two parts: the delay section and weighting section. Each of the parts is built around a Non-Volatile-Memory element. In the delay section, an input spike arrives as the input at time t_{in} of the circuit to the gate voltage of a n-type transistor (READ terminal). The pulse pulls down the voltage of a capacitor (Vcap),

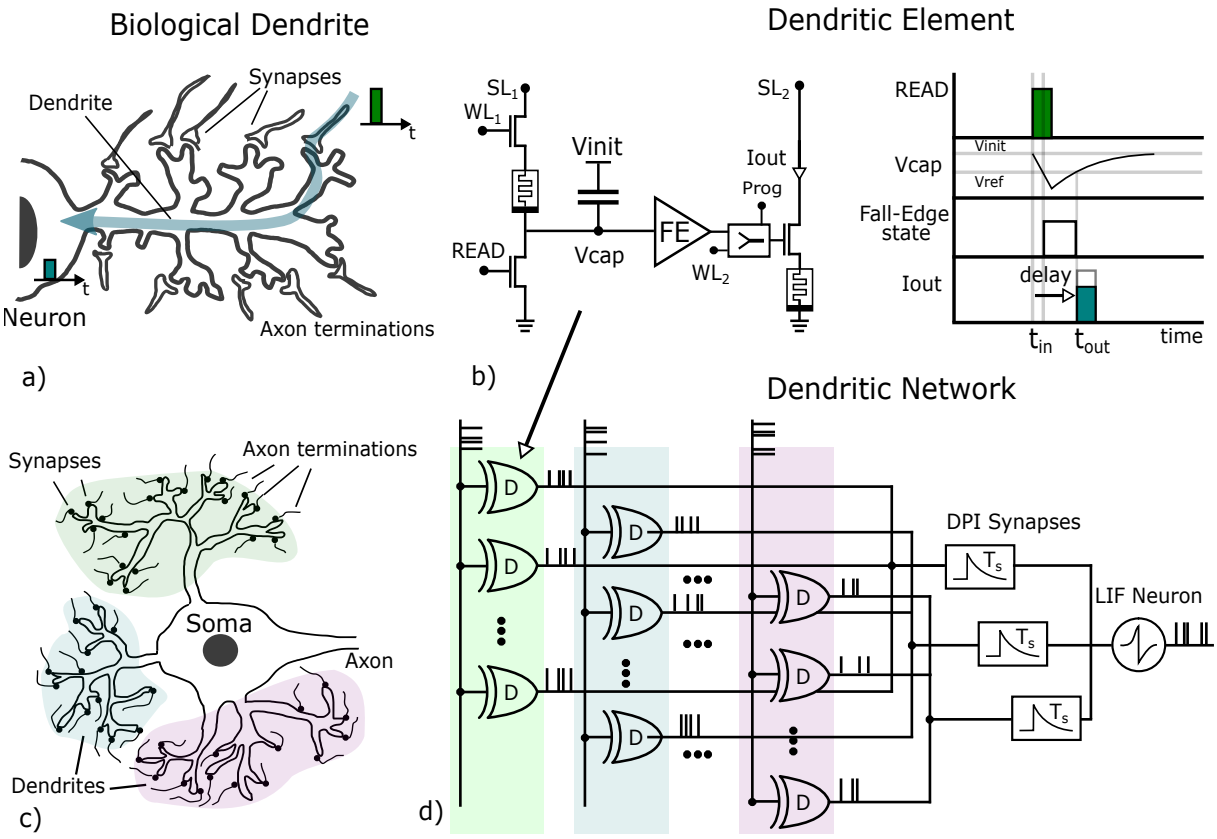


Figure 5.7: Dendritic computation and Dendritic circuit. (a) Depiction of a biological Dendrite, receiving different synapses from different axon terminations. Each dendrite subdivides into smaller dendritic spines. Inputs in the form of action potential come from the axons and transmit through synapses, where they are modulated in magnitude, and are further processed in the Dendritic branch before reaching the Neuron. (b) Dendritic Circuit scheme and its basic working as a function of time. The circuit features 2 RRAMs, with Delay and Weight functions, a Capacitor and a Fall-Edge detector. c) Neurons can feature many dendritic branches, each of which connecting to many synapses. d) This inspired the Dendritic Network, formed by several Dendrite Circuits, grouped into Dendritic branches macro-circuit. The branches' output are processed by their dedicated DPI synapse and ultimately stimulate a LIF Neuron.

which is in parallel to the Delay-NVM. Note that, except for the programming phse, the SL terminal of the Delay-NVM is shorted to the voltage on the capacitor Vinit. Depending on the resistance of the delay-NVM, the capacitor voltage Vcap relaxes back to the resting potential Vinit with different time constant. $\tau_{delay} = R_{del} \times C$ the time constant depends on the delay-NVM's resistance (R_{del}) and on the capacitance C. The temporally decaying voltage of the capacitor is connected to a Fall-Edge (FE) detector. The FE detects the Vcap potential crossing the reference voltage Vref, at time t_{out}. When this happens, the FE circuit emits a spike in the output, being delayed in time respect to the input READ

pulse (Fig. 5.7b). Here starts the second section of the circuit, consisting of a simple 1T1R device, referenced as weight-NVM. The voltage pulse coming from the FE opens the gate of the weight 1T1R, drawing a current Iout proportional to the conductance of the NVM. This current is the output of the circuit and constitutes a delayed and weighted current version of the input pulse presented on the READ terminal.
In biology, multipolar neurons in the cortex develop into multiple dendritic branches which receive a large number of inputs (Fig. 5.7c). Each of the branches is highlighted with a different color and generally. Mimicking such biological structure, the Dendritic Network (Fig. 5.7d) also shapes into different dendritic branches (highlighted with different color each) all insisting on the same output neuron. In this network, the same spike train is in shared across a dendritic branch, which is a collection of N dendritic elements. M branches can be stacked together in parallel receiving as many input signals. DPI synapses can be interposed between the dendritic branches and the output neuron to dynamically process the incoming trains of spikes. The output LIF neurons thus has to its disposal a highly processed version of the inputs and a lot of degrees of freedom to adapt for a given task. The aim is to expand the computational power of the single neuron and in turn of neuromorphic neural network.

5.2.2 Assessing the computational power of Dendritic Networks

Dendritic Networks promise to boost the computational capacity of neuromorphic neural networks. On one side, dendrite expand the dimensionality of the inputs, on the other they provide more align the useful temporal information from an input. This last aspect is potentially disruptive. Conventionally, neural networks make use of recurrent connections to form complex temporal dynamics that allows to make sense of long temporal input sequences. Such recurrent dynamics are hardly trained with the back-propagation techniques. It is highly desirable to avoid complex dynamical systems that are hard to adapt for a given task. In the deep learning realm, recurrent neural networks have been overcome by attention-based models [298]. The ideal behind attention layers is to learn the importance of the temporal features from a sequence, and to pass them to the following layers. In this sense, dendritic branches perform similar computation. Each element in a branch produces a delayed version of the input and, by learning the synaptic weights, one selects which combination of delays transmit the most relevant features of the input spike train to the output neuron.
To assess the computational power of such concept, a simple experiment is performed. Figure 5.8a shows a single dendritic branch fed by a Random input pattern. The input spike train is delayed and weighted by N dendritic elements and then integrated by an output Leaky-Integrator neuron. Such neuron type does not spike and only integrated input spike onto the membrane voltage with leakages. The task is to reproduce a signal generated by summing 3 sinusoids with periods $[\tau/1.5, \tau/2, \tau/2.5]$, with τ being the length of the target signal. In the dendritic element, delays are initialized from a distribution

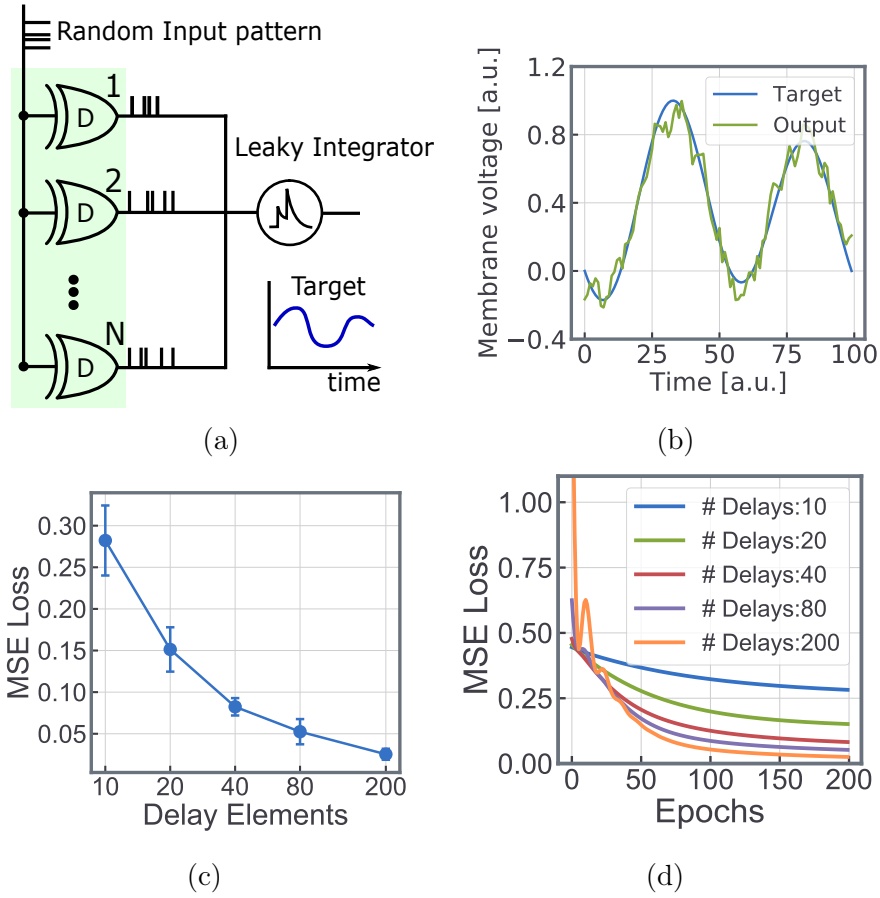


Figure 5.8: Dendritic Network benchmarked on a pattern regression task. (a) A random input patter is fed to the dendrite, depicted as a collection of dendritic elements in green. Each of the element delays input spikes of about $t_{del,i}$ and weighs the delayed train of spikes by $w_{del,i}$. The resulting delayed and weighed trains of spikes are linearly accumulated at the output neuron, in this case a Leaky-Integrator (LI) neuron. Note that this neuron does not feature a threshold and never spikes. This neurons' membrane voltage is compared to an arbitrary time-varying signal. For this task, the target signal is a sum of sinusoids with given periods. (b) An example of successful learned signal. The generated patter is depicted in blue and represents a sum of 3 sinusoids with periods equal to $\tau/1.5$, $\tau/2$ and $\tau/2.5$, where τ is the sequence length. The green signal is the membrane voltage of the LI output neuron and matches the blue target, as requested by the task. (c) Performance in the patter reconstruction task as a function of the number of delay element in a dendrite. MSE stands for Mean Squared Error between the target and the output is selected. As intuitive, a larger dendrite allows for more information to be available at the output neuron, increasing the expressivity of the network. (d) The importance of the number of delay elements is confirmed by the behavior of the loss as a function of training iteration, for different dendrite sizes. Smaller dendrites saturate the capacity of the network way earlier in the training than larger ones, which can reconstruct the target patter better.

and fixed afterwards. In this experiment, the distribution from which delays are picked is a simple uniform distribution in a range from 0 to the length of the target signal. Weights are instead learned with gradient descents. Fig. 5.8b shows one such target signals as well as the output from the neuron's membrane voltage which closely follows it. Matching the target signal means that the task was correctly performed. Learning to reconstruct generated signals is intuitively easier if more dendritic elements are at disposal: this is proven in Fig. 5.8c in which the Mean Squared Error (MSE) is plot against the number of delay elements in the branch. Fig. 5.8d analyses the behavior of the MSE loss along the training epochs, in which smaller dendritic branches saturate the capacity of the network much sooner in the learning phase than bigger networks.

Application to a real task: ECG classification The dendritic network is aimed at solving real world tasks and this is why it is tested with the ECG dataset, already mentioned in Chapter 1. This also allows for a direct comparison with a RSNN model. The dendritic network used for this test is depicted in Figure 5.9a. The ECG signal is delta-modulated and yields 4 input spike trains. Each of them is fed to each own dendritic branch having N dendritic elements. Lastly, the delayed and weighted versions of the spikes are integrated by the output neuron, whose activation determines the presence (or absence) or arrhythmia. As a first experiments, the 4 dendritic branches are assigned fixed delays drawn from a narrow distribution centered around different mean delays. Each dendritic branch contains 10 elements. This permits to identify the most useful range of time constants needed to solve the task. Figure 5.9a reports the results of such experiment where the error rate in the test set of the ECG dataset is plotted as a function of the delay range in the dendrites. Centering the delays around 160 ms results in the best performance, despite acceptable error rates are obtained with most delay ranges. Too short delays don't increase enough the amount of information available at the neuron, while too long delays only transmit the information after the neuron is asked to classify the input. Furthermore, unnecessary delays result in a temporal lag for computation which is undesirable. The analysis on the importance of the different delays allowed to calibrate a hardware-plausible version of the Dendritic Network in which Ferro-electric Tunnel Junctions (FTJ) devices and RRAMs are assumed as the NVM technology of choice in the architecture and the capacitor size is tailored for the application. FTJs are chosen are the delay NVMs as they reach mpressive high resistances, well over 10 Gω i the High Resistive State [299]. RRAMs work as the weight NVMs due to their ease in programming and multi-level capabilities [129]. To reach delays on the order of 100 ms, capacitance on the order of C $1pF$ are required. A hardware plausible dendritic network is formed by drawing delays from the HRS distribution of resistance of FTJs, as in [300, 299]. Weights are first trained with floating-point-32 (FP32) precision and later converted to RRAMs. Results are presented in Fig. 5.9c: FP32 based dendritic network reaches an error rate of about 4.7%, which is increased marginally when converting to RRAMs. The resuls are compared to a conventional RSNN with 128 neurons in the hidden layer. Error

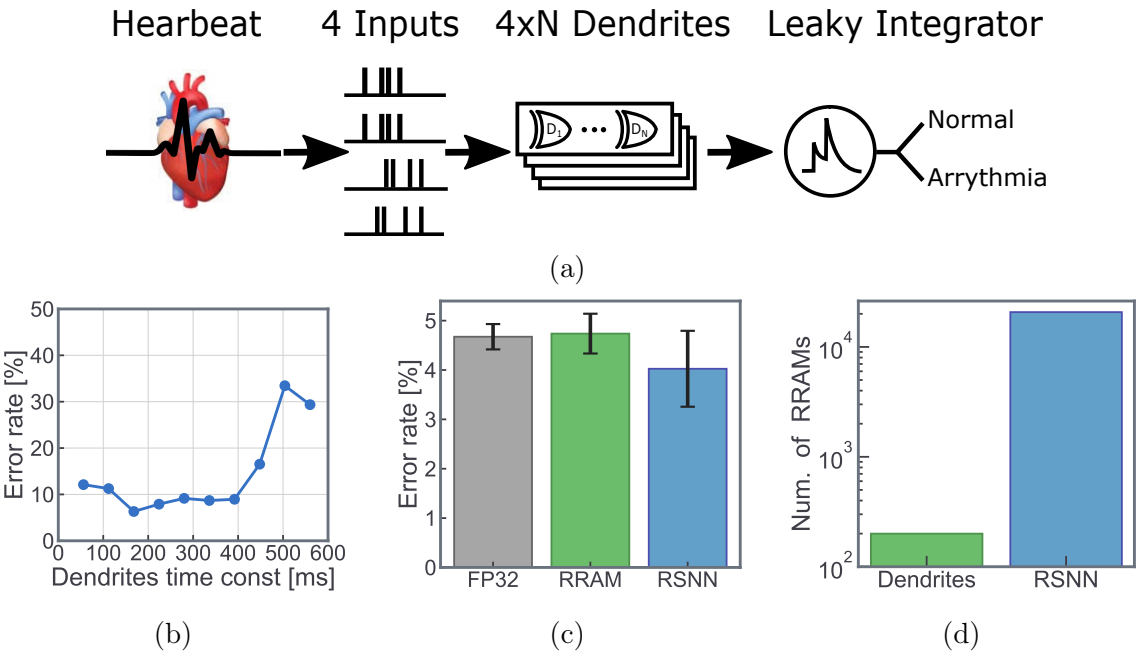


Figure 5.9: **Dendritic Network employed to detected ECG arrhythmia.** (a) ECG is recorder by 2 electrodes and converted in 4 trains of spikes. The 4 inputs are all fed to Dendrite circuits, each with N dendritic elements. The dendritic branches converge onto an output neuron, a Leaky Integrator. The output neuron receives weighted and delayed versions of the 4 inputs channels, and increases its membrane voltage when an a case of arrythmia is presented. (b) Considering Dendrites with 10 elements each, the delayed introduced by the dendritic elements are sampled around a mean value, in the x-axis of the plot. The accuracy of the network is evaluated for each of the delays, discovering that a delay of around 150 ms is optimal to solve the ECG task. (c) The performance of the Dendritic Network with 4 branches and 128 dendritic elements per branch - in the cases of a pure software model (FP32) and hardware implementation (RRAM) - is compared to an RSNN with 128 neuron in the recurrent layer, solving the ECG task. The two architectures reach comparable error rates. In the Dendritic Network, conversion to RRAM results in a minor accuracy loss. The RSNN, due to its much higher parameter count, reaches a slightly higher accuracy. (d) Number of parameters - or RRAMs - employed in the network. The Dendrites-based network makes use of $D_{In} \times N_{del}$ RRAMs, D_{In} being the input dimension, N_{del} the size of the dendrites. RSNN, instead, use N^2 parameters, N being the number of neurons in the recurrent layer (omitting the input and output layers for simplicity). N=25 for the case under consideration.

rate in the RSNN results slightly lower, even though it depends on the number of neurons in the hidden layer. A smaller RSNN would results in lower accuracy. The size of 128 is chosen as it matches that of the delay elements in the dendritic network. Because both neurons in RSNN and delay elements in a the dendritic network feature a capacitor, they result occupying comparable sizes. However, Dendritic Network result having much lower

memory footprint (MF) than RSNNs. The MF of a RSNN is calculated as:

$$MF_{RSNN} = D_{in} \times D_{hid} + D_{hid}^2 + D_{hid} \times D_{out} \tag{5.1}$$

where $D_{in,hid,out}$ are the input, hidden and output layer dimensions. For the RSNN of choice, the MF is of 18,176. For the Dendritic Network the MF is evaluated as follows:

$$MF_{dendrite} = 2 \times D_{in} \times D_{elements} \tag{5.2}$$

where $D_{elements}$ are the number of dendritic and elements, the factor 2 is beacure there are 2 NVMs per element. With 25 dendritic elements only, the dendritic network MF is of just 200, 2 orders of magnitude less than that of the RSNN. These values are shown in Fig. 5.9d.

5.2.3 Hardware implementation of the Dendritic Network

Dendritic Network have proven useful from the algorithmic point of view, but neuromorphic engineering is about building powerful bio-inspired hardware solutions. The advantages of the dendritic circuit element are translated to hardware, making use of Non-Volatile-Memories. In this section, the dendritic element circuit is presented in a version featuring 2 RRAM devices as NVM elements. As discussed previously, RRAM are gret at implementing the synaptic weight of the dendritic circuit element, due to their low power programming operations and non-volatility. For the delay NMV element, the main design requirement is for the resistance of the NVM - coupled with the Capacitor - to produce a large enough time constant, useful for computation. RRAMs can achieve up to 1 $G\Omega$ of resistance in the HRS and even higher resistances in the pristine phase. Optimization of the stack composing the RRAM can further improve this high resistance. However, Ferro-electric Tunnel Junction [301, 299] devices are more adapted to reach higher resistance level, comfortably above the 10GΩ mark. In ferroelectric RAMs, information is encoded as the ferroelectric polarization of an insulating layer sandwiched between two electrodes. The polarization of the device, and its resistance in turn, can be modified with the application of an electric field across the electrodes. Conventionally, ferroelectric devices yield tiny readout currents. However, when the ferroelectric layer is thin enough, an FTJ device is formed by allowing a tunneling current to flow and to produce large ON/OFF resistance ratio, up to 10^3 [300]. The ferroelectric can be build based on HZO_2 oxide topped by a 1.5 nm, sandwiched between two TiN layers [302]. The integration process is compatible with that of RRAMs, making the two technologies possibly co-integrated in a chip. The combination of FTJs and RRAMs would benefit from the large resistance of FTJs for the delay NVM element and from the flexibility of the LRS of RRAMs. For the following, both NVM memories will be referenced as delay and weight RRAM, since the circuit was originally developed around this technology. However, the memory elements can be swapped with FTJ devices without modification of the surrounding circuit elements. As the programming operation is carried out outside of the chip, no dedicated circuit is calibrated to either of the two memristive technologies.

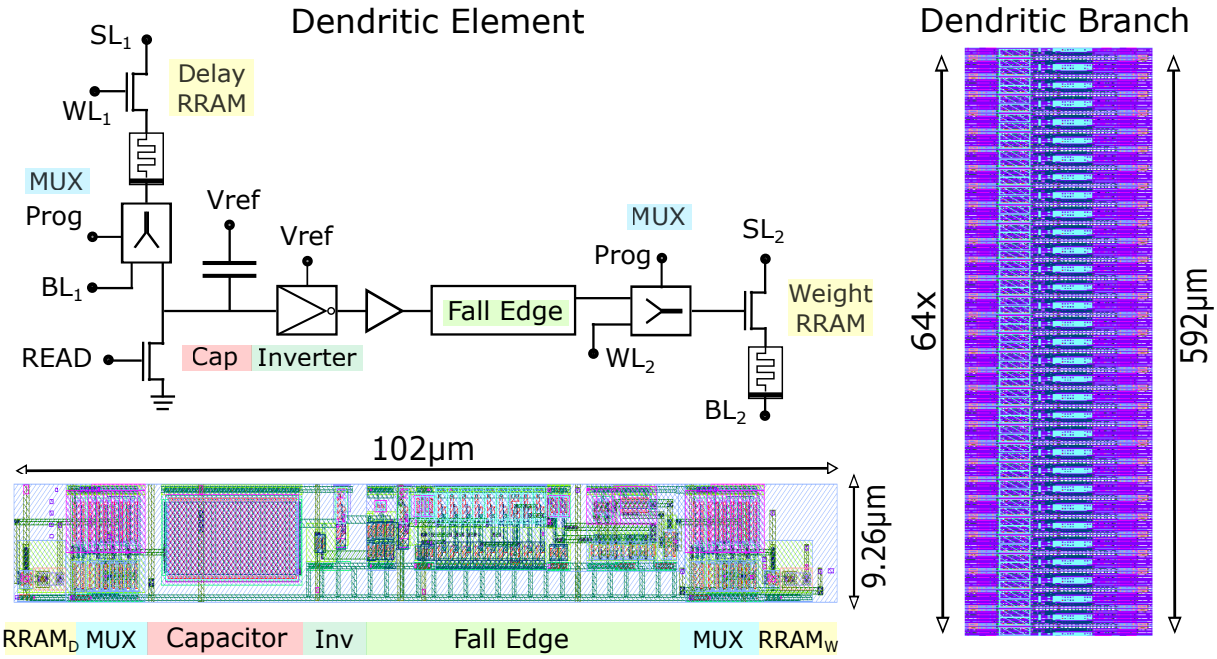


Figure 5.10: Dendritic circuit and its physical layout. (a) Detailed circuit scheme with all the circuit components. (b) Physical layout of the dendritic circuit, with the components placement highlighted below. c) View of 64 Dendritic circuit forming a Dendritic branch.

Dendritic circuit element The Dendritic circuit element is introduced in Fig. 5.7 is analyzed here in more details. Figure 5.10 reveals the main components of the circuits: the Delay and Weight RRAMs, MUXes, a Capacitor, an Inverter and a Fall-Edge detector. Multiplexers are necessary to decouple RRAM devices with the circuit during the programming phase. The MUX are indeed controlled by a voltage terminal called Prog, which is high (at 5V) when the given RRAM is selected for programming. This removes the SL/WL terminals contacts (for the delay and weight RRAM respectively) from the rest of the circuit and creates a connection with an external pad, operated by a dedicated machine for the programming operation. If the circuit is built into an array (as seen on the Dendritic Branch on the right) these terminals are handled by a peripheral circuit which selects which RRAM to connect to the pad so to be programmed.
When the Prog voltage is off, the circuit operates as a dendrite. Inputs of the circuit are presented as voltage pulses at the READ terminal, pulling a current from the capacitor and causing a voltage depolarization. During the functioning of the circuit, the SL terminal is connected to the Vref voltage, being the reference or resting voltage of the capacitor. This puts the delay RRAM in parallel with the capacitor and in that way the voltage across it will relax back to Vref with a time constant $\tau = R_{RRAM} \times C$. The inverter after the capacitor has a switching voltage of Vref/2 and changes states both after the arrival of the READ pulse and in the relaxation phase of the capacitor potential. The Fall Edge detector block is devoted to capture the latter change of state and produce an output

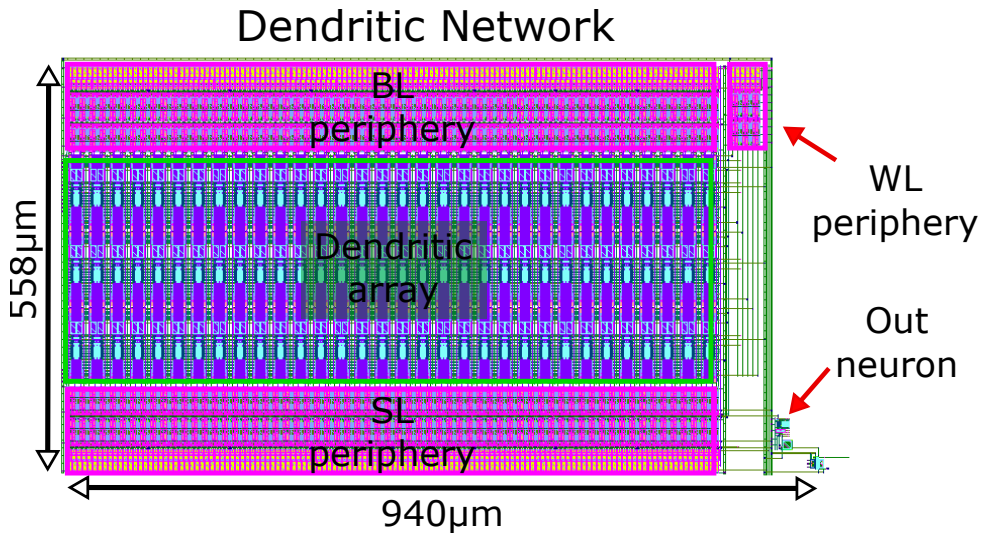


Figure 5.11: Layout of a 64-elements, 3-Branches Dendritic Network. The Dendritic array features all the dendritic circuit (63x3 in this case). BL/SL periphery are sized according to the dendritic circuits, while the WL periphery is placed on the right and is much smaller, having to manage 6 columns of devices only. The output neuron appears on the right.

pulse when the capacitor voltage has crossed the inverter's threshold in the relaxation phase. The Fall-Edge detector is also equipped with a pulse extender which assures the correct pulse-width for all the spikes emitted by the dendritic element. Output pulses are modulated in amplitude through the weight RRAM. The output section of the circuit is composed of a 1T1R device, whose SL is fixed at the reading voltage, while the bit line is normally set as ground. A current proportional to the conductance of the weight RRAM is generated when the voltage pulse from the Fall-Edge opens the word lines of the 1T1R. A circuit similar to that described in Chapter 1 can be used to read the current and buffer it to a neuromorphic circuit, like a LIF neuron.
Figure 5.10 shows the physical layout of the circuit with the components' placement highlighted below. The size of [102x9.26]μm is dominated by the Capacitor and the Fall Edge detector. These are the area where some improvements can provide the most benefits in terms of density. Also, it is reminded that the circuit represents a first prototype for the Dendrite element concept and presents large margins for improvements in term of density. The right part of Figure 5.10 shows the Dendritic Branch circuit featuring 64 dendritici circuit elements. It size is [592x102]μm.

The dendritic circuit element is the building block of a Dendritic network. Several deendritic elements can be stacked on top of each other and form a dendritic branch, which in this implementation share the inputs. Several branches are juxtaposed next to each other to form a dendritic network. In the simplest form, a dendritic layer of a dendritic network is formed by placing a neuron at the output of all the dendritic branches. However the concept can be expanded by forming a neural network where each neurons features a

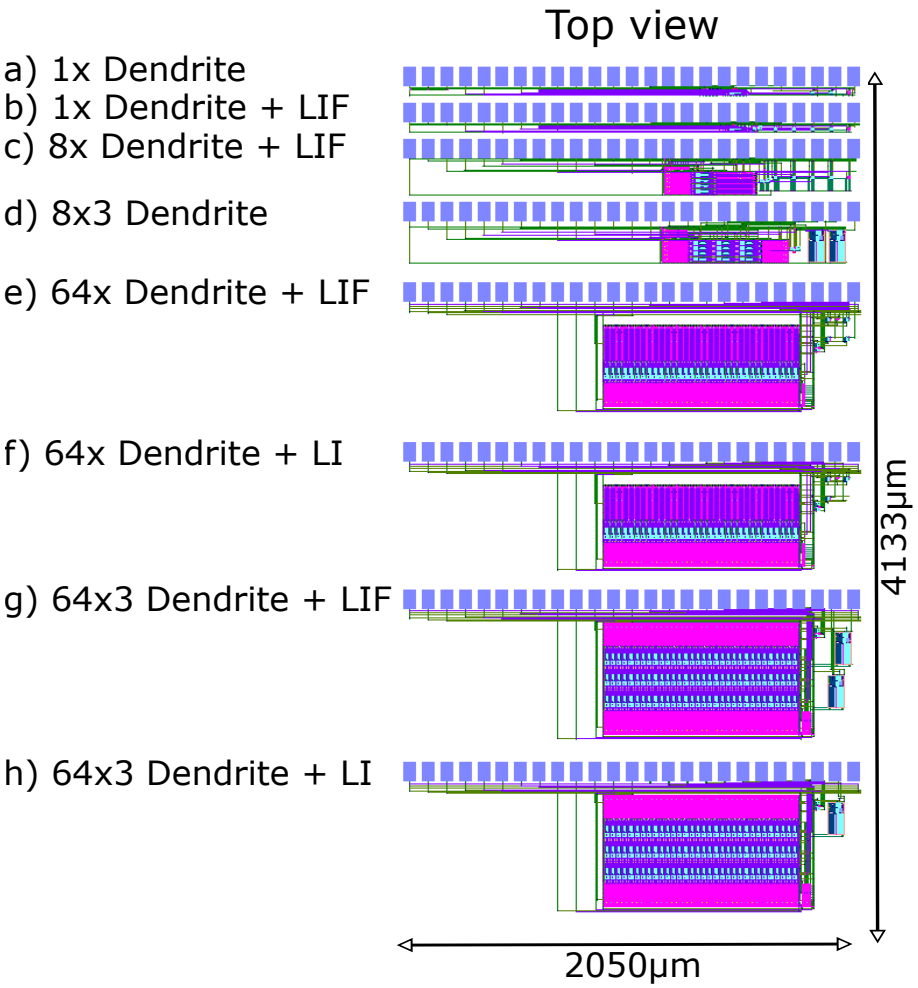


Figure 5.12: MAD200V6, Dendritic circuit Contribution. 8 circuits have been designed with different versions and sizes of the Dendritic Network. a,b) feature a single Dendrite Circuit, with a LIF output neuron. c) is a Dendrite Branch of 8 elements. d-h) represent different sizes of Dendritic Network (AxB, A number of dendritic elements, B number of branches).

dendritic network, with multiple dendritic branches. Layers can be stacked one next the other, with the benefit of producing even larger delays and non-linear computation.
Figure 5.11 shows the construction of a Dendritic Network. Several dendritic elements are sharing the weight-RRAM's SL (SL_2) and the READ terminal, delivering the input. Each RRAM in the dendritic branch can be programmed independently by acting on the selected combination of source/bit line and word lines. During the programming mode, each source/bit line is treated independently and handled by a SL/BL periphery, of the same type of those presented in Chapter 2 The periphery of the dendritic array is a scan chain which connects one selected source/bit line to an output pad to program the selected 1T1R device. Word lines are share along the dendritic branches, the small

purple block on the upper right part of Figure 5.11. The SL periphery features a switch that connects all the source lines from the weight-RRAM to a common node. This node is feeding the currents read from the weight-RRAMs to the output neuron, on the lower right portion of Figure 5.11. Several dendritic branches are grouped together to form a Dendritic Network. The output of the synapses is integrated by the output neuron. Effectively, this network replaces and augments the functions of a crossbar array performing simple Matrix-Vector-Multiplication, endowing the array with dynamical processing properties. The circuit shown in Figure 5.11 represents a 64x3 Dendrite network, formed by 3 branches with 64 elements each. The final size of this circuit is [940x558]μm.

Different versions of the dendritic network concept have been designed and included in the test chip. Simpler structures like the 1x Dendrite (Fig. 5.12a) feature a single dendrite circuit element, and are aimed at thoroughly characterize the potential of the circuit. Progressive size increases should allow to study the effect of scaling up the network size. Detrimental effect such as IR drop and capacitive effects due to the long wires in the array might negatively impact the results of the circuit. Both Leaky-Integrate and Leaky-Integrate-and-Fire neurons are features in the circuit. The first is designed to solve regression tasks 5.2.2, where a target signal is mapped onto the membrane voltage of the neuron. LIF neurons can instead be used for classification, like in the case of the ECG task 5.2.2. Overall the controbution includes 8 circuits and fits into an area of [2050x4133]μm

5.2.4 Extensions of the project

This section represents the first step into the development of dendritic, analog, memristive-based neuromorphic systems. The good results shown in 5.2.2 are convincing arguments for dedicating further attention and efforts in dendritic computation. Not only the circuit can be optimized in footprint-area, as mentioned, there are changes and improvement that can be beneficial to the concept. Here is a list of the main points for developing the dendritic network concept.

- Improving footprint-area
- Expanding the Dendritic Network to multiple layers
- Learning the delays to optimize computation
- Endowing the circuit with on-line learning capabilities

Each of this point is a possible research direction that can take the dendritic computation to the next level.

Improving the footprint area Footprint area is the main issue of the dendritic network, in its current realization. Improvements can be made on two sides. First, the layout of the circuit can be substantially optimized. An area improvement of 4/5x can be achieved

with clever design. Second, technology can help scale down the dimension of the circuit. The most obvious improvement is to chose a more advanced technological node than 130 nm. This would shrink the size of the peripheral circuit and of the Fall-Edge detector. Unconventional technologies might enable a more radical scaling of dimensions. Volatile memristors [157] could replace capacitors - the largest elements in the architecture - and achieve orders of magnitude of improvements in footprint-area.

Expanding the Dendritic Network to multiple layers A possible weak point of the dendritic circuit is that is produces delays that are limited to the time constant of the circuit. However, if one chains multiple dendritic elements, the output spike would be delayed by the sum of the individual delays at each element. This would allow to obtain a wider range of delays that are possibly key to solve complex tasks. Furthermore, one could design a SNN where each synapse features a delay. In this network, the dendrite arrays would replace conventional synaptic memristive arrays. Such multilayer dendritic network would endow SNN with unprecedented computational power, avoiding the need of recurrent connections which make the training phase harder.

Learning the delays to optimize computation In the examples in this sections, delays of the dendritic networks are set a priori, and not a trainable parameter of the network. One improvement over this learning scheme would be to learn such delays so to optimize them for the given task. Learning of synaptic delays is proven effective in these works [303, 304].

Endowing the circuit with on-line learning capabilities In the current form, the dendritic network is a fixed architecture during inference mode. However, it is desirable that both delays and weights are adapted to the incoming inputs, in a continual learning fashion. Chapter 3 introduced the Delta Rule learning circuit: such circuit would fit to adapt the weight memristor. However, a circuit that acts on the delays is to be developed.

Chapter 6

Discussion

This thesis groups different projects about RRAM-based, bio-inspired, in-memory computing. Despite being presented in separate chapters, they constitute a united vision of neuromorphic memristive systems. Multiple technical aspects have been considered, from sensory signal treatment to in-memory computing architectures, from circuit design to device characterization. Chapter 2 begins with an off-line learning algorithm to correct hardware defects in memristive spiking neural network hardware implementations. The algorithm is called neuromorphic calibration procedure and mitigates RRAM variability and analog circuit mismatch. Surprisingly, the neuromorphic hardware calibration procedure turns the defective mismatched analog circuits into a positive feature for computation, boosting the performance compared to an ideal SNN with perfect circuits. The variability of RRAMs is thoroughly investigated and classified into three categories: relaxation, retention and read-to-read. Each of these terms affects performance in different ways. However, relaxation has the largest impact in the short-term. Retention problems mainly decrease classification accuracy on complex tasks, read-to-read makes each inference slightly different. Compared to a processor featuring identical circuits but a digital communication protocol and digital weights, the RRAM-based SNN is one order of magnitude more energy efficient in communicating spiking events through the chip. Stimulated by these results, a RRAM-based SNN is built around a 32x32 array, with dedicated peripheral circuits for programming and accessing the devices. A custom-made peripheral circuit hosts Leaky-Integrate and Fire neurons and differential-pair-integrator synapses, thus making the circuit a 32 neurons recurrent spiking neural network. The circuit takes inputs in the form of voltage pulses and each of the 32 neurons can be routed to an output read-out pad via a selector. However, scaling up this architecture goes against a few challenges such as IR drop and undesired capacitive effects. The Mosaic concept is proposed to solve these issues and design large-scale RRAM-based SNNs. Mosaic is a systolic array composed of neuron and routing tiles, the latter being RRAM arrays dedicated to connecting neurons across the large mesh of tiles. The architecture has been demonstrated effective at solving

benchmark tasks and Mosaic is proven to make more efficient use of memory elements compared to a single array of the same size.
Similar circuits to those described in the RRAM-based SNN are deployed to perform object localization with auditory cues. Chapter 3 presents a bio-inspired system in which two innovative technologies from CEA Leti are combined: piezoelectric-micromachined-ultrasound-transducers as sound sensors and resistive memories as computational substrate. Ultrasound sensors are thoroughly characterized and paired with a dedicated neuromorphic treatment of their signal. Inputs from the sensors are translated to the spike domain and then processed by a computational map, inspired by that found in the barn owl. Thanks to its biologically inspired data treatment, the system is 4 orders of magnitude more energy efficient than a conventional signal processing algorithm run on a commercially available micro-processor.
Chapter 4 also presents the concept of a system, in this case for artificial olfaction. Silicon photonics sensors record the presence of olfactory stimuli in the environment. The sensors are sensitive to operating conditions and react differently to each gas. Furthermore, gas sensors suffer from slight integration process variation, thus require tailoring the computational model for gas classification to the specific use case. On-chip learning is proposed as the solution in the form of the Delta learning Rule. A circuit is developed to work with a RRAM array. Computer-in-the-loop experiments and system-level simulations assess the performance of the learning procedure. The delta rule is found to be a good compromise between the efficacy of the on-chip training and the complexity it adds to the architecture.
Most neuromorphic systems are based around neurons and synapses, and plasticity is featured only in the latter elements. Chapter 5 proposes novel ideas to go beyond this scheme and expand the functionalities of neuromorphic architectures. In the memristive-self-organized-network, neurons are endowed with plasticity thanks to enhanced circuits featuring RRAM devices. LIF neurons express intrinsic plasticity which is combined with spike-driven-synaptic-plasticity to form an unsupervised learning rule that boosts the performance of spiking neural networks in temporal tasks. The learning rule is designed to cope with RRAMs variability, and results proved that performance even increased when variability is introduced, compared to when ideal devices are considered. A thorough analysis of the role of variability and noise in the network confirmed that RRAMs can be used successfully as plastic elements in neurons. In the second section of the chapter, a dendritic circuit is presented, performing two main functions in spiking neural networks: weighting and delaying spikes. These two features are enabled by RRAM devices, thanks to which both weight and delays are controlled. Dendritic networks are formed when each input is fed to several dendritic circuits. At first, single-layer dendritic networks are tested on simple benchmark tasks, though deeper networks can be designed for more challenging tasks. The dendritic network is proven to reduce the memory requirement in SNNs, reaching accuracy comparable to that of RSNNs.

6.1 Perspectives on future work

While the projects in this thesis seek to find solutions and propositions in the context of neuromorphic in-memory computing, they leave space for future improvements and expansions. Perspectives on the works are grouped into three main points and discussed below.

Hardware developments With the surge of ASIC accelerators for Edge AI, it appears that a solution for energy and area-efficient computation has already been found and belongs to the digital domain. Indeed, the performance per watt of digital accelerators has substantially increased in the last 10 years. However, it remains to be seen if the trend will continue over the next 10 years. Some of the improvements in digital ASICs are certainly due to ameliorations in the architecture or of the neural network models they implement, but a large step in performance and efficiency is due to the improved technology nodes. Evidence is pointing toward a slowed-down continuation of Moor's law [67], and the next generation of digital ASICs is likely to use only marginally better transistors. Even reasoning at the economical level is not reassuring: will Edge AI systems gain enough scale in production to justify the costs of adopting the latest 2 nm technology [65]? If digital ASICs will reach a plateau in performance in the next decade, the analog in-memory community has to be ready to propose a paradigm shift. However, some key aspects of analog in-memory circuits have to be drastically addressed in the meantime.

- **Reading memristors efficiently**: the key advantage of NVMs is that they don't require static power consumption and some of them store multiple bits per cell. To extract this information, however, complex circuits are conventionally employed. Minimizing the circuit complexity and energy consumption to read the devices is important to enable the potential of NVMs. Samsung [97] and UCSD University [98] have proposed accelerators for IMC exploiting clever designs to read the conductive state of MRAMs and RRAMs respectively. Solutions of the same kind are required for neuromorphic chips as well.

- **Minimizing footprint area**: neurons and synapses take a large share of area in neuromorphic processors [29, 31]. This has to be reduced as large area results in high costs of production for a chip. The key is to replace or shrink the large capacitors in neuromorphic circuits [23, 70] with either innovative devices or with advancements in the algorithms. In the first case, the aim is to develop an integrated device in the back-end-of-line that features high capacitance, perhaps exploiting high-k dielectrics or novel materials. In the second case, the need of large capacitance could be avoided at the algorithm level, finding new computational primitives that are effective on temporal processing.

- **Scaling up network size**: deep learning has shown that scaling up neural networks produces novel interesting behavior and improves the performance [99]. While

neuromorphic systems are more interesting to edge applications - thus with energy and footprint-area constraints - network size still plays a big role in increasing performance. After all, the brain makes use of more than 80 B neurons, yet it consumes around 20 W [185]. However, scaling is only possible if the two bullet-points above (reading NVMs efficiently and reducing footprint area) are solved. The Mosaic architecture presented in Chapter 2 is a promising candidate to explore the advantages of scaling, but it would be advisable to adopt 3D architectures in the future to enable higher degrees of connectivity. New process integration technologies could be the solution for future neuromorphic chips.

Bio-inspired network architectures As much as the hardware, neuromorphic algorithms also require improvements to compete with existing deep learning models. The aim is not to replace state-of-the-art generative models with hundreds of billions of parameters [99], but to be able to express the same performance - or more - with comparable hardware resources. Looking at deep learning, several network architectures have been proposed. In temporal sequence processing, recurrent neural networks have been updated into LSTMs [289] and GRUs [290], and now completely replaced by Transformers [298]. This evolution has not happened for neuromorphic networks, which are stuck with recurrent spiking neural networks architectures. An update is strongly required or neuromorphic will soon be obsolete. Changes can be configured in two aspects:

- **Architecture**: changing the connectivity of neurons, perhaps inspired by biology, to improve the performance. Particularly, inspiration could be drowned from insects, the simplest forms of neural systems. Dalgaty et al. [194, 186] proposes a neural network with fixed sparse connectivity to improve the performance of the network, while utilizing less parameters. Chapter 3 proposes a computational model inspired by the auditory pathways of the barn owl to perform object localization. The system reaches orders of magnitude greater energy efficiency compared to a conventional signal processing algorithm.

- **Computational primitives**: chapter 5 presents the dendrite circuit in dendritic networks, reducing the memory footprint in SNNs and still performing on-par with RSNNs. Delays are introduced in SNNs to improve the temporal processing capabilities of the network and to avoid the need for recurrent pathways, typical of RSNNs. Dendrites could represent a good candidate as a computational primitive that elevates neuromorphic networks. Nonetheless, there is still a lot to find out in the brain that can be transferred to neuromorphic chips: biological neurons are much more complex than their neuromorphic counterparts. Computational neuroscience will hopefully identify which elements in biological neurons are expressing such high computational power.

Improved on-chip learning While impressive steps forward have been made in neuromorphic learning algorithms that are compatible with the hardware and that can be

performed on-chip, a major shift in paradigm is necessary. The classical scheme of supervised learning where labelled input data is provided in the learning phase does not match the way learning occurs in humans and is also limited in edge applications. The operating conditions at the edge can change throughout the life of a neuromorphic system, and the chip needs to be able to adapt. In conventional supervised learning, human intervention is necessary to provide labels, but this constitutes a great limitation. On the other hand, novel learning schemes need to adapt to the continuously evolving computational substrate in neuromorphic chips, specifically when including novel integrated devices, like non-Volatile-Memories. New learning procedures need to exploit the non-linear and dynamical properties of analog electronics and NVMs, including the their mismatch and variability. The following three points represent the main research paths in neuromorphic learning.

- **Self-Supervised learning**: Self-supervised learning is a novel technique relying on the data itself to produce labels, making the neural network acquire a model of the input data without external supervision. This has boosted the performance of deep learning models, as it allowed to scale up datasets without the need to produce manual labels. The same effect is expected for neuromorphic computing, which could leverage more data from neuromorphic sensors to build larger datasets. More importantly, without the need for labels, the model could learn from the stream input data collected while operating at the edge. Early example of self-supervised neuromorphic learning have been proposed [123], yet there's still a lot to be developed and discovered.

- **Meta-Learning**: it is difficult to imagine gradient descent methods fully implemented on memristive architectures on-chip, due to the high variability of the devices. One strategy is to prepare the model for on-chip learning with off-line training, optimizing the forthcoming in-situ weight updates. This is among the benefits of Meta-Learning, a training paradigm that aims at optimizing the learning procedure itself. Similarly to Transfer Learning, the model is pre-trained off-chip, however meta-learning involves optimizing the weight updates as well. The goal is to pre-train the model and provide a meta-initialization of the parameters so that it is then easy to update the weights to solves a particular task, despite the variability introduced by the devices.

- **Continual Learning**: learning new tasks and adapting to the environment is useless if previously acquired knowledge is forgotten. The well known problem of catastrophic forgetting [305] hinders edge AI system to perform on-chip learning when shifting between tasks in a sequential manner. Solutions have been proposed to mitigate this problem, such as weight consolidation and meta-plasticity [122]. The theory is that once a network is optimized for one particular task, learning a new one should not disrupt the weight configuration. Weight updates should account for the consolidation of important parameters in the network storing the information

from previously acquired knowledge.

Chapter 7

Appendices

SNN hyper-parameter tuning

Spiking neural networks are computational models in which time plays a key role. Neurons and synapses are dynamical elements that process the input spikes they receive at a speed that depends on their time-constant. Given that the physical implementation of neurons and synapses is made with analog electronic circuits, the time constant is due to a capacitor that integrates input stimuli and that leaks the accumulated charge through a leakage element. In the LIF neuron and DPI synapse, a transistor biased with a sub-threshold gate voltage modulates the leakage rate and thus the time constant. At the circuit level, the time constant indicates the memory window of the single circuit or, more technically, the time it takes for the voltage of the capacitor to decrease to about $e^{-1} \approx 0.37$ of the initial value. Evidently, the time constant of neurons and synapses in a spiking neural network influence the response at the system level. As a rule of thumb, the time constant of the circuit should be matched to the temporal features the network is asked to classify of process. For example, the pronunciation of spoken digits between zero and nine normally lasts for about 1 second, so a dynamical network analysing such inputs should feature components that get close to this time-constant. However, technological limitations in capacitor size and transistor leakage current constrain the time constant of analog circuit. A detailed analysis has been carried out to find out which time constant in neuron circuits maximises performance at the network level in the different training procedures described in Section 2.3. The Neuromorphic Calibration Procedure (NHC) features analog circuit mismatch and RRAM variability in both training and inference phases, Homogeneous SNNs have ideal - and thus homogeneous - circuit behavior in both training and inference, Non-Calibrated SNNs are trained on homogeneous circuits and testes with circuit variability. The characterization results of neurons and synapses performed in Figure 2.2 are used in the NHC simulation for hardware-aware training. Results are summarized in Figure 7.1a,b,c.

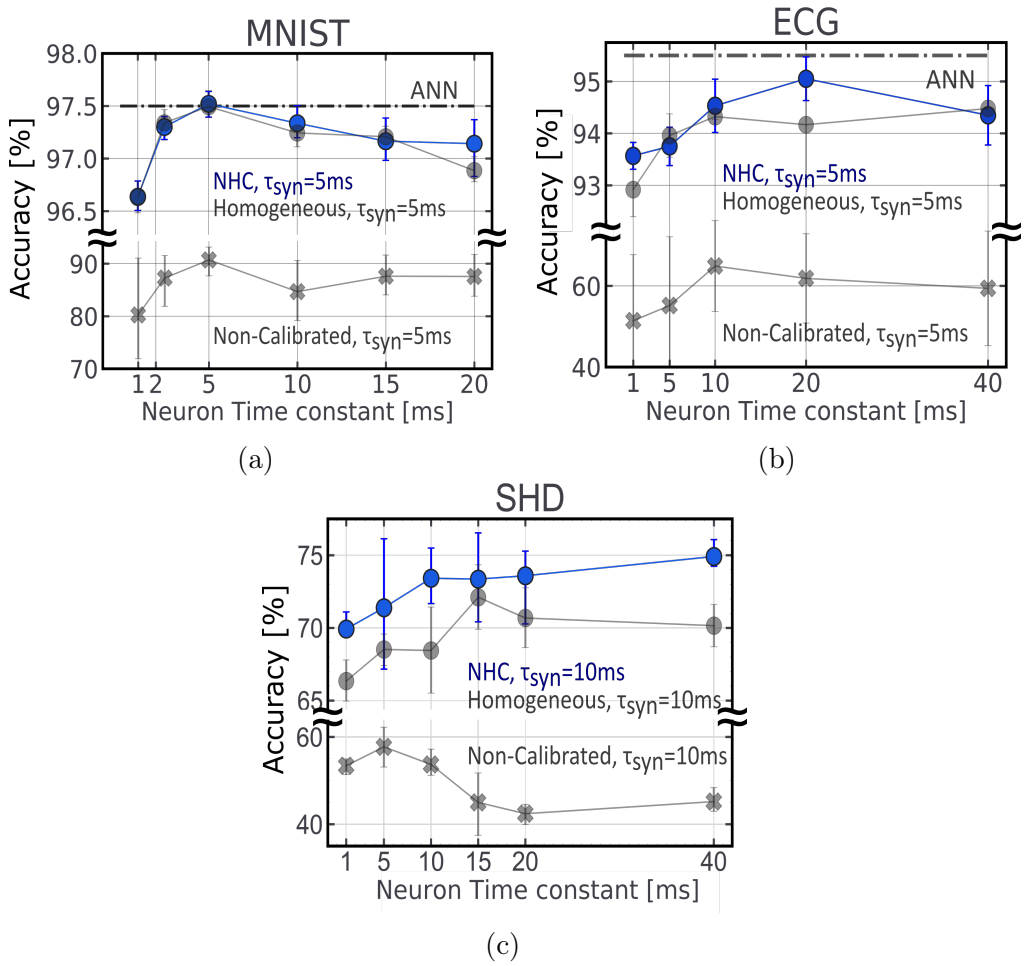


Figure 7.1: Accuracy over MNIST (a), ECG (b) and SHD (c) analysed as a function of the neuron time constant. For MNIST, which does not involve temporal features in the dataset, time constant of the neurons does not have a great effect and performance is close to the ANN reference (horizontal dashed line) as long as the time constant is greater than 1 ms. ECG and SHD tasks demand to process the data in the temporal domain, so the time constant has a greater effect on accuracy. ECG requires 20 ms, SHD 40 ms, even though the trend is positive with the time constant and performance could improve further. However, silicon circuits are limited to the rang shown in the figure. As said in Chapter 2, the effect of the NHC procedure is to boost performance respect to the Homogeneous case, while Non-calibrated SNNs perform poorly.

As the MNIST task does not feature relevant temporal content, being static images, the neuron time constant only modestly influences performance. In fact, the MNIST dataset is transformed to the spike domain and each pixel intensity is assigned a spike time in a certain interval, which is set at 100 ms in this experiment. The conversion from intensity and time-to-spike is linear and only 1 spike is emitted per pixel. This means that the only temporal factor in the input data is the duration of the stimulus. In other neuromorphic

vision datasets, like the DVS 128 [306], inputs are recorded by a DVS camera and thus capture the natural evolution of the stimuli in time. Still, classifying MNIST does not require special dynamical properties at the network level and the NHC and Homogeneous cases almost match in performance. Evidently, the Non-Calibrated case has poor accuracy as the circuit mismatch introduced in the test phase by the analog circuit is not accounted for during training.
The ECG and SHD tasks feature more interesting temporal stimuli and require dynamical processing to reach satisfactory performance. ECG's data are sourced from the MIT-BIH dataset and sub-sampled to include the 4 most represented types of arrhythmia, along with "normal" heartbeats. Each class has about 1000 data-points, for a total of 4802. 70% of the dataset constitutes the training set, 10% the validation set (used for fine-tuning the model) and 20% as a training set. ECG data-points present a heart-beat from channel 2 of the MIT-BIH recording, last for 700 ms and are centered around the peak - said R peak - of the voltage recording. Arrhythmia manifests as an anomaly in the heart-beat rhythm and in irregularity of the normal phases that composed the signal (P-Q-R-S-T). The simulation results confirm that temporal dynamics is more important in the ECG task and optimal performance are obtained with a time constant for neurons of 20 ms. Increasing or decreasing the temporal response of the circuit - and network in turn - decrements the classification accuracy. Notably, the NHC procedure yields the best results, higher that the Homogenous case. This could be explained by the increased dynamics of the network with variability in analog circuits, each of which features a slightly different time constant. This heterogeneity in the network can be exploited with gradient-descent technique which optimize the network given the hardware constraints. As usual, the Non-Calibrated case yields poor performance.
The spiking Heidelberg Dataset is a collection of recordings from a wide number of speakers of spoken digits from zero to nine, in both English and German. Digits pronunciations last for about 1 second. To solve this task, models need to process the temporal information in the recording. It has been shown that high time constants lead to large classification accuracy [35, 149]. In the simulations in Figure 7.1c, results are monotonic with the time constant, confirming the literature. Higher time constants are not included in the analysis as not technologically plausible with the analog circuits designed in this work. As the temporal features in the SHD dataset are even more relevant that for SHD, the advantage of heterogeneity is even more evident. Non-Calibrated procedure fails at solving the task with satisfactory accuracy levels.

Characterization of RRAMs and Neuromorphic circuits

The experimental results presented in this thesis are all been performed in CEA Leti testing facilities and with a setup that is described in this section.

Measurement setup

Experimental setup for the tests of circuits presented in this thesis is divided into two main categories: tests of RRAMs (both single devices and arrays) and tests on analog custom circuits. In both cases, wafers in either 200 mm and 300 mm format are handled with an automatic chuck holder, featuring step motors that can be commanded by a computer. The chuck holder hosts the wafer and brings it in contact to a set probes, composed of 25 metallic connectors arranged in a line. The probe is 2 mm in length and it matched the dimension of the pads with which the chips are fabricated. The chuck holder can be programmed to switch between dies, when large-scale measurements are to be performed. To measure a test structure in a single die, the setup described in Figure 7.2 is employed.

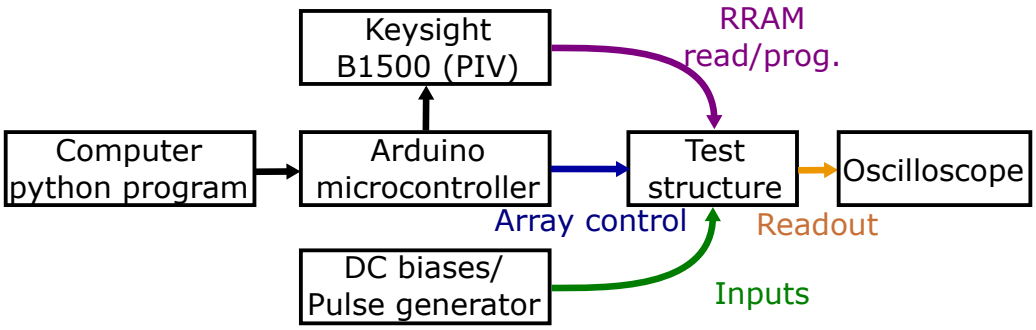


Figure 7.2: Measurement setup scheme. A computer controls the whole experimental setup with a python script. An Arduino micro-controller receives the commands from the computer and performs the addressing of the RRAM array in the Test Structure. At the same time, the Arduino times the action of a Keysight B1500 machine [257]. In particular, pulsed-current-voltage (PIV) units from the machine are used to read and program the RRAMs in the array. Pulse generators and DC voltage and current biases control parameters of the analog circuit and provide the inputs. While the circuit is working, its outputs are recorded on a oscilloscope, which digitizes the measured traces.

A computer runs a python script with a program that controls most of the instruments in the setup. If necessary, the program also executes specific algorithm needed in the test: it is the case for Smart Programming for RRAMs and the Delta Rule algorithm in Computer-in-the-loop experiments. The computer commands an Arduino Mega2560 micro-controller, which has the role of performing the addressing of RRAM arrays, interacting with the peripheral circuits shown in [reference]: the interaction is highlighted with the blue arrow. Not only, the Arduino also commands a Keysight B1500 [257], used to operate the RRAM devices (purple arrow in Fig. 7.2). To do so, the three Pulsed-Current-Voltage (PIV) outlets are connected to the three terminals of a 1T1R device and custom pulses - controlled by the python program - are applied. Conventionally, the bit-line of the device reading terminal for the READ, FORM and SET operation. For RESETs, positive voltage pulses are applied to the bit line and the resulting current is read at the source line. In SET and FORMING operations, positive voltage pulses are applied to the source and

word lines. Pulses are squared with a standard duration of 1μs, though RRAMs have been demonstrated to require much shorter pulse-widths per operation [75]. Currents read by the PIVs are digitized and loaded on to the computer. Outputs from the test structures are instead read by an oscilloscope and later saved on a USB stick.

RRAMs can be analysed with the quasi-static programming method, in which a slowly varying voltage is incrementally increased and decreased, controlling the current flowing through the device. During the whole procedure, the word line is kept open, at 4.8V. Results of such operation on a stand-alone 1T1R RRAM device are reported in Fig. 7.3. For the positive side of the plot (SET and Forming operations), a positive voltage is applied at the Top of the device (source line), while the Bottom (bit line) is kept at ground. The reverse happens for the left side (RESET). The device under test is in the Pristine state, meaning that no conductive filament has been formed already. Following the green line, the device exhibits high resistivity up to reaching 1.7V $V_{TOP-BOT}$ where a conductive filament starts forming and the device is *Formed*. At that point, the device is in the High Conductive State (HCS). Inverting the sign of the voltage, the conductive filament is disrupted at around -1V and the device switches to the Low-Conductive-State (LCS). Applying a new cycle of positive voltage, the blue curve, the device can be SET when reaching a voltage of 0.7V, returning to the HCS.
The switching voltages of 1T1R cells depend on the programming strategy and on the transistor size. For the quasi-static method, switching voltages are generally lower, as the programming conditions are applied for long times. For pulse-based methods, switching voltages are traded off with latency and RRAMs can be programmed in this manner in less than 100 ns [75]. In general, RRAM devices require switching current on the 10μA order of magnitude. Transistor size changes the voltage applied to the gate, modulating the current allowed through the device. Such current is called compliance current.

Neuromorphic circuit measurements

Neuromorphic circuits are analog electronic circuits operating in the sub-threshold regime and with biased transistors. In order to operate correctly, they require inputs in the form of voltage pulses - representing spikes - and DC biases both as voltages and as currents. CEA Leti experimental laboratories feature HP/Agilent 8110a programmable pulse generators, that are used to provide the input spikes. Continuous currents and voltages are provided by simple DC sources. Outputs are samples by an oscilloscope.

Delta Modulator The Delta Modulator circuit converts analog input signals into trains of spikes, with minimal information loss and low power-consumption. Inputs are encoded in two spike trains: the UP channel encodes increments of the inputs signal, DW channels encodes the decrements. Combining the information from the UP and DW channels, the input signal can be approximated, less for the DC component which is not encoded.

The circuit in Figure 7.4a has been designed by Thomas Dalgaty and it features two

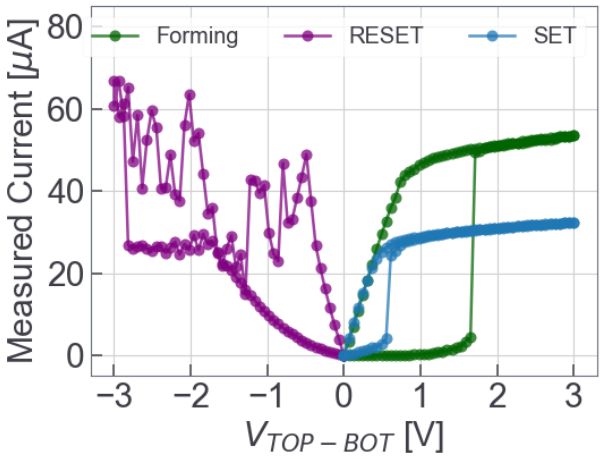


Figure 7.3: RRAM quasi-static characterization. A RRAM in the pristine state is subject to a voltage applied to its Top and Bottom terminals ($V_{Top-Bot}$): when this voltage overcomes the Forming threshold, a conductive filament is formed and the device passes to the High-Conductive-State (HCS). Applying a negative $V_{Top-Bot}$ weakens the filament, up to the RESET threshold at which the filament is disrupted: the RRAM is then in the Low-Conductive-State (LCS). By applying a positive voltage again, the conductive filament is restore, that being a SET operation.

stages. In the first stage, the input signal is filtered with a capacitor C_1 which removes the CD component and passes the signal to two comparators. When the signal increments by more that $V_{thr,UP}$, the upper comparator is triggered and an output spike is produced in the UP channel. If the signal then decreases by more that $V_{thr,UP}$, a spike is emitted in the DW channel. In the second stage of the circuit, the two output channels are fed back to an OR logic gate which, when active, emits a pulse charging the capacitor C_{freq}. This capacitor leaks charge through the p-MOS transistor controlled by V_{freq}, modulating the leakage rate and thus the time constant of the capacitor C_{freq}. During the period in whic C_{freq} is loaded, an n-MOS transistor clamps the input node to ground, making the Delta Modulator unresponsive. This limits the activity of the circuit, and thus the power consumption. The circuit is produced with 130 nm technology and measured with the described setup. Two example of the functioning of the circuit are provided in Figures 7.4b and 7.4c. In Figures 7.4b, an input voltage of 750 mV peak-to-peak voltage, offset of 500 mV and frequency of 100 Hz is fed to the Delta Modulator circuit, controlled by a biased $V_{freq} = 1V$. Given the slow oscillations of the input, the circuit is producing a high number of output spikes in both UP and DW channel during the period of the signal. Increasing the input frequency of the input to 500 Hz and keeping V_freq fixed, the stage two of the delta modulator circuit prevents from emitting more output spikes. This results in a lower number of emitted spikes in a period of the input stimulus, and a consequential loss of information.

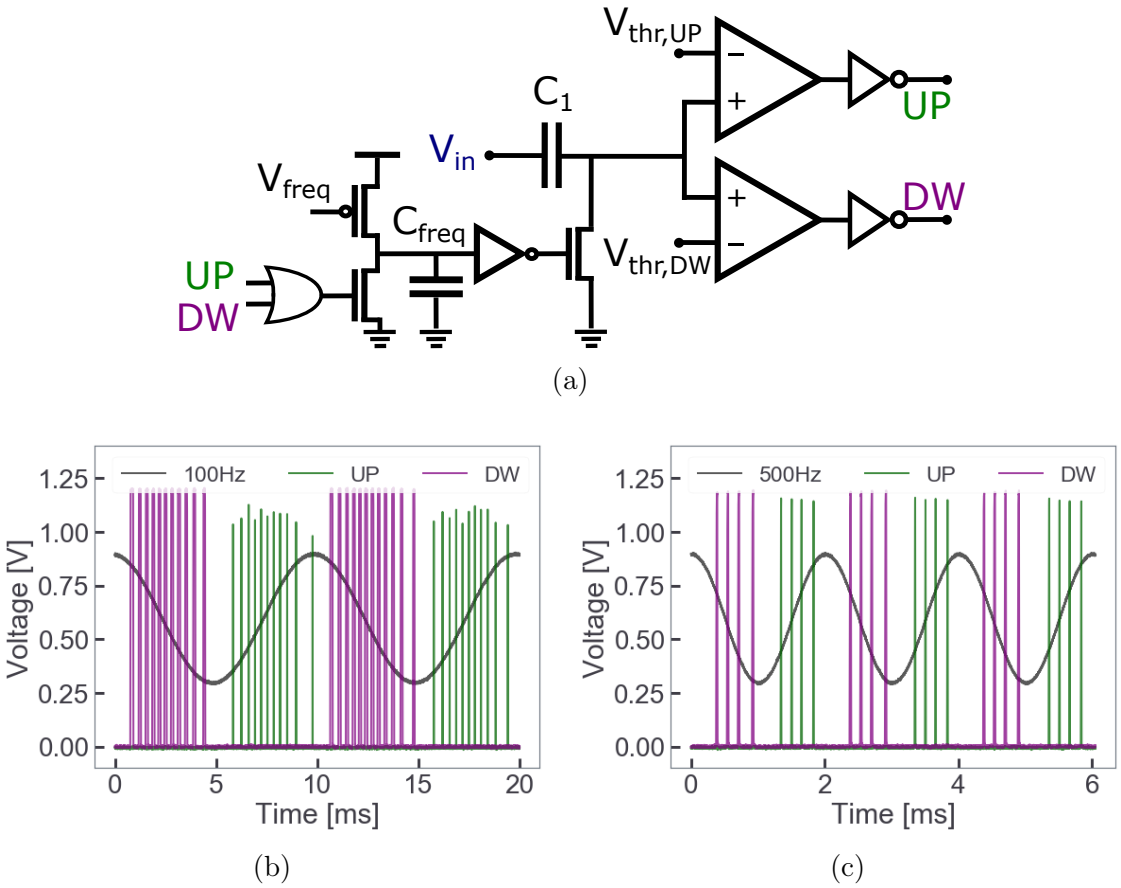


Figure 7.4: Delta Modulator circuit and its measurements. (a). Delta Modulator circuit schematics, as designed by Thomas Dalgaty. In input signal Vin is passed through a decoupling capacitor, which removes the DC voltage component. When either of the two output comparator is activated, with input voltage being greater then $V_{thr,UP/DW}$, a spike is emitter at either UP or DW. This spike is fed back in the circuit and charges a capacitor C_{freq}, which - when charged with a voltage higher than the switching threshold of its neighbouring buffer - clamps the input node to ground. This hinder the generation of spikes for a time controlled by V_{freq} which determines is the leakage rate of the capacitor C_{freq}. In this way, the Delta Modulator can be made more or less responsive to inputs, adapting the spike conversion to incoming stimuli. (b) Measurement of the circuit with an input at 100 Hz, recording both input and outputs from an oscilloscope. Given the slowness of the input, several spikes are emitted by the circuit, making for a good representation of the input information. (c) Similar measurement but for a 500 Hz input, keeping the same bias condition for the delta modulator circuit. This time, the circuit hardly follows the input oscillations, producing less spikes at the output, thus partially losing its information.

Leaky-Integrate and Fire (LIF) neurons are analog electronic circuits presented in Figure 2.2. Measurements presented in this figure shown the dynamical behavior of the circuit,

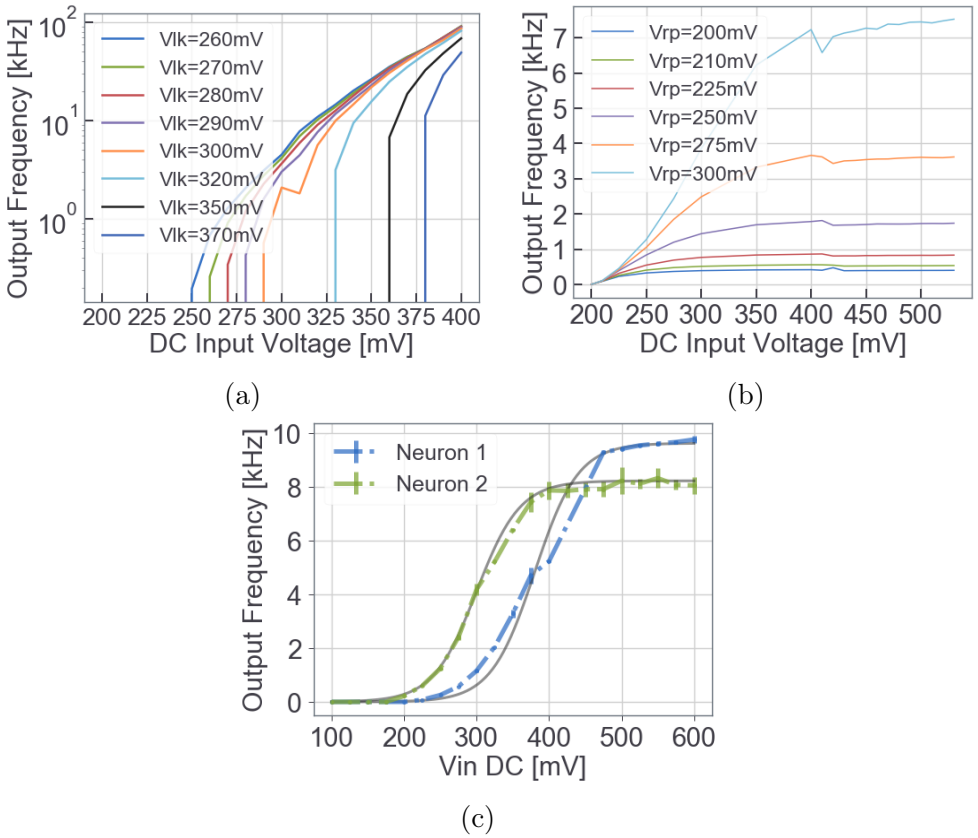


Figure 7.5: Characterization of a DPI neuron. The input voltage Vin is swept in a range of voltages and the output frequency is recorded with an oscilloscope. (a) The leakage parameter Vlk is modified during the test, modulating the leakage rate of the membrane capacitor C_{Mem}. Higher leakages result in less spikes and thus lower output frequencies. (b) Analysis on the refractory period, controlled by the bias Vrp. When Vrp is low, the leakage rate of the refractory capacitor C_{refr} is higher and the refractory period lower. This means that the neuron remains inactive for less time, and it can fire more frequently. During this analysis, the Vlk bias was set at 260 mV. (c) By combining the action of the Vlk and Vrp biases, the neuron exibists a sigmoid function for the output frequency depending on the input DC voltage. The sigmoidal behavior is reproduced by 2 neuron circuits, highlighting the variability between analog circuits.

which integrates input pules on the membrane capacitor C_{mem} and emits output spikes when the voltage at the capacitor overcomes the switching voltage of an inverter in the output stage. The circuit features the leakage transistor controlled by Vlk, modulating the time constant, and the refractory period transistor Vrp, controlling the refracory period. Experiments have been performed to analyse the input-output behavior of the circuit, with an input DC voltage and measuring the output spike frequency. Figure 7.5a analyses the effect of changing Vlk in the range $[260mV - 370mV]$, while Vrp is fixed at

450 mV. The circuit is not capable of producing output spikes with too low input DC voltage. The onsed of output spikes varies depending on Vlk: higher leakage rate require stronger input magnitude to elicit output spikes. In general, output spike frequency tends to converge when the input becomes stronger, as the input current becomes much stronger than the leakage current. Figure 7.5b reports a similar analysis in which Vrp is swept in the $[200mV - 300mV]$ and Vlk is fixed at 260mV. Again, output spikes are not produced when the input is weak. Output frequency saturates to values that depend on the Vrp: higher biases make the refractory period shorter and thus allow the neuron to spike at higher frequencies. Combining these two biases, a sigmoidal input-output relationship can be obtained. This is shown in Figure 7.5c, in which two neurons belonging to two different dies are tested with Vlk=300mV and Vrp=300mV. The different shape in the two traces highlights the variability between sub-threshold analog circuits.

Delta Rule on CIFAR10

CIFAR10 is a modestly challenging dataset that is tamed with end-to-end training of convolutional neural networks. In the framework of on-line learning, additional constraints are imposed, such as memory requirements of the embedded accelerators. Ideally, learning on-line is performed with a batch-size of 1, meaning that the updates to the weights occur after each input streamed to the network. This is convenient to minimize memory requirements and energy consumption too, however on-line learning can also be performed accumulating the weight update over multiple input data-points. The delta rule, optimizing the last layer of a neural network only, does not require to store the activations of all the neurons into memory. For this reasons, learning over batches of input data-points is feasible with the delta rule. The requirements to do so are that the activations of the last layer of the neural network model of choice have to be stored to memory and so do the outputs. For the case of EfficientNet trained on CIFAR10, the 1280 activations of the last hidden layer and the 10 outputs should be saved. Considering 8-bits activations, the memory requirement is $10,320 \times B$ bits, where B is the batch-size. To optimize the parameters, activations are used to compute the activation, a step that results in large energy consumption [307]. The Delta Rule circuit relaxes this requirements by directly calculating the weight update in the analog domain. Batched learning with this circuit can be performed by directly averaging the weight updates for each parameter, with the updates ΔW_{ij} that have to be stored into memory. For 4-bits weight this results in $51k \times B$ bit, a larger memory footprint, but with the advantage of having already computer the weight update.
The hardware realization of a circuit storing the weight updates values and performing the programming operation on-chip is beyond the scope of Chapter 4. To assess the importance learning with batched data-points, a experiments trains the last layer of EfficientNet over the CIFAR10 dataset with the Delta Rule. Variability and Relaxation are disabled for this experiment.

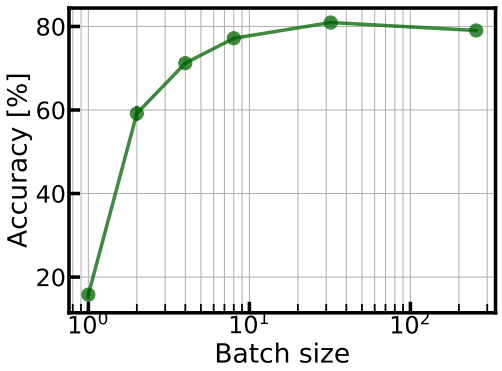


Figure 7.6: Effect of batch-size when training on the CIFAR10 dataset with the Delta rule. To yield good accuracy, a batch-size of 8 is required.

Clearly, low batch-size results in unacceptably low classification accuracy on the test-set. Performance are satisfactory starting from a batch-size of 8. For the experiments in Figure 4.9b and Figure 4.9c a batch-size of 32 is utilized.

Bibliography

[1] Filippo Moro et al. "Neuromorphic object localization using resistive memories and ultrasonic transducers". In: *Nature Communications* 13.3506 (2022), pp. 2041–1723. DOI: https://doi.org/10.48550/arXiv.2202.05094.

[2] Melika Payvand et al. "Self-organization of an inhomogeneous memristive hardware for sequence learning". In: *Nature Communications* (2022).

[3] Filippo Moro et al. "Hardware calibrated learning to compensate heterogeneity in analog RRAM-based Spiking Neural Networks". In: *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2022. DOI: https://doi.org/10.48550/arXiv.2202.05094.

[4] Thomas Dalgaty et al. "Hybrid neuromorphic circuits exploiting non-conventional properties of RRAM for massively parallel local plasticity mechanisms". In: *APL Materials* 7.8 (2019), p. 081125. DOI: 10.1063/1.5108663. eprint: https://doi.org/10.1063/1.5108663. URL: https://doi.org/10.1063/1.5108663.

[5] Yiğit Demirağ et al. "PCM-Trace: Scalable Synaptic Eligibility Traces with Resistivity Drift of Phase-Change Materials". In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401446.

[6] Thomas Dalgaty et al. "Synapse circuit for three-factor learning". In: 17454435. 2022.

[7] Thomas Dalgaty et al. "Circuit and method for spike time dependent plasticity". In: 17454373. 2022.

[8] Carver Mead. "How we created neuromorphic engineering". In: *Nature Electronics* 3.7 (2020), pp. 434–435. ISSN: 2520-1131. DOI: 10.1038/s41928-020-0448-2. URL: https://doi.org/10.1038/s41928-020-0448-2.

[9] Giacomo Indiveri. "Neuromorphic Engineering". In: *Springer Handbook of Computational Intelligence*. Ed. by Janusz Kacprzyk and Witold Pedrycz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 715–725. ISBN: 978-3-662-43505-2. DOI: 10.1007/978-3-662-43505-2_38. URL: https://doi.org/10.1007/978-3-662-43505-2_38.

[10] Giacomo Indiveri and Timothy Horiuchi. "Frontiers in Neuromorphic Engineering". In: *Frontiers in Neuroscience* 5 (2011). ISSN: 1662-453X. DOI: 10.3389/fnins.

2011.00118. URL: https://www.frontiersin.org/articles/10.3389/fnins.2011.00118.

[11] John Von Neumann and Ray Kurzweil. *The computer and the brain*. Yale university press, 2012.

[12] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

[13] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[14] Marvin Lee Minsky. *Computation*. Prentice-Hall Englewood Cliffs, 1967.

[15] Kunihiko Fukushima et al. "An electronic model of the retina". In: *Proceedings of the IEEE* 58.12 (1970), pp. 1950–1951.

[16] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[17] David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*. Vol. 1. IEEE New York, 1988.

[18] Misha Mahowald and Carver Mead. "The silicon retina". In: *Scientific American* 264 (May 1991), pp. 76–83.

[19] R.F. Lyon and C. Mead. "An analog electronic cochlea". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36.7 (1988), pp. 1119–1134. DOI: 10.1109/29.1639.

[20] Misha Mahowald and Rodney Douglas. "A silicon neuron". In: *Nature* 354.6354 (1991), pp. 515–518.

[21] M.A. MAHOWALD. "Chapter 15 - Evolving Analog VLSI Neurons". In: *Single Neuron Computation*. Ed. by THOMAS MCKENNA, JOEL DAVIS, and STEVEN F. ZORNETZER. Neural Networks: Foundations to Applications. San Diego: Academic Press, 1992, pp. 413–435. ISBN: 978-0-12-484815-3. DOI: https://doi.org/10.1016/B978-0-12-484815-3.50023-2. URL: https://www.sciencedirect.com/science/article/pii/B9780124848153500232.

[22] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. "A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 43 (Mar. 2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[23] Giacomo Indiveri et al. "Neuromorphic Silicon Neuron Circuits". In: *Frontiers in Neuroscience* 5 (2011). ISSN: 1662-453X. DOI: 10.3389/fnins.2011.00073. URL: https://www.frontiersin.org/articles/10.3389/fnins.2011.00073.

[24] Ben Varkey Benjamin et al. "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations". In: *Proceedings of the IEEE* 102.5 (2014), pp. 699–716.

[25] Mike Davies et al. "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning". In: *IEEE Micro* 38.1 (2018), pp. 82–99. DOI: 10.1109/MM.2018.112130359.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[27] Stanisław Woźniak et al. "Deep learning incorporating biologically inspired neural dynamics and in-memory computing". In: *Nature Machine Intelligence* 2.6 (2020), pp. 325–336.

[28] *Scopus - discover the most reliable, relevant, up to date research*. URL: https://www.scopus.com/search/form.uri?display=basic#basic.

[29] Saber Moradi et al. "A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)". In: *IEEE Transactions on Biomedical Circuits and Systems* 12.1 (2018), pp. 106–122. DOI: 10.1109/TBCAS.2017.2759700.

[30] Alexander Neckar et al. "Braindrop: A Mixed-Signal Neuromorphic Architecture With a Dynamical Systems-Based Programming Model". In: *Proceedings of the IEEE* 107.1 (2019), pp. 144–164. DOI: 10.1109/JPROC.2018.2881432.

[31] Christian Pehle et al. "The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity". In: *CoRR* abs/2201.11063 (2022). arXiv: 2201.11063. URL: https://arxiv.org/abs/2201.11063.

[32] Henry Markram et al. "Introducing the Human Brain Project". In: *Procedia Computer Science* 7 (2011). Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 11), pp. 39–42. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2011.12.015. URL: https://www.sciencedirect.com/science/article/pii/S1877050911006806.

[33] Filipp Akopyan et al. "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.10 (2015), pp. 1537–1557. DOI: 10.1109/TCAD.2015.2474396.

[34] Charlotte Frenkel et al. "A 0.086-mm^2 12.7-pJ/SOP 64k-Synapse 256-Neuron Online-Learning Digital Spiking Neuromorphic Processor in 28-nm CMOS". In: *IEEE Transactions on Biomedical Circuits and Systems* 13.1 (2019), pp. 145–158. DOI: 10.1109/TBCAS.2018.2880425.

[35] Charlotte Frenkel and Giacomo Indiveri. "ReckOn: A 28nm Sub-mm2 Task-Agnostic Spiking Recurrent Neural Network Processor Enabling On-Chip Learning over Second-Long Timescales". In: *2022 IEEE International Solid-State Circuits Conference (ISSCC)*. Vol. 65. IEEE. 2022, pp. 1–3.

[36] Alessandro Aimar et al. "NullHop: A Flexible Convolutional Neural Network Accelerator Based on Sparse Representations of Feature Maps". In: *IEEE Transac-*

tions on Neural Networks and Learning Systems 30.3 (2019), pp. 644–656. DOI: 10.1109/TNNLS.2018.2852335.

[37] Chang Gao, Tobi Delbruck, and Shih-Chii Liu. "Spartus: A 9.4 TOp/s FPGA-Based LSTM Accelerator Exploiting Spatio-Temporal Sparsity". In: *IEEE Transactions on Neural Networks and Learning Systems* (2022), pp. 1–15. DOI: 10.1109/TNNLS.2022.3180209.

[38] S Peioglou-Harmoussi et al. "F-response frequency in motor neuron disease and cervical spondylosis." In: *Journal of Neurology, Neurosurgery & Psychiatry* 50.5 (1987), pp. 593–599.

[39] Simon Thorpe. "Spike arrival times: A highly efficient coding scheme for neural networks". In: *Parallel Processing in Neural Systems and Computers* (Jan. 1990).

[40] Vincent Chan, Shih-Chii Liu, and Andr van Schaik. "AER EAR: A Matched Silicon Cochlea Pair With Address Event Representation Interface". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.1 (2007), pp. 48–59. DOI: 10.1109/TCSI.2006.887979.

[41] Alberto Mazzoni et al. "Morphological Neural Computation Restores Discrimination of Naturalistic Textures in Trans-radial Amputees". In: *Scientific Reports* 10.1 (Jan. 2020), p. 527. ISSN: 2045-2322. DOI: 10.1038/s41598-020-57454-4. URL: https://doi.org/10.1038/s41598-020-57454-4.

[42] A. Egea-Weiss et al. "High Precision of Spike Timing across Olfactory Receptor Neurons Allows Rapid Odor Coding in Drosophila". In: *iScience* (2018). DOI: doi:10.1016/j.isci.2018.05.009.

[43] Shih-Chii Liu et al. "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms". In: *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2010, pp. 2027–2030. DOI: 10.1109/ISCAS.2010.5537164.

[44] Chiara Bartolozzi, Giacomo Indiveri, and Elisa Donati. "Embodied neuromorphic intelligence". In: *Nature Communications* 13.1 (Jan. 2022), p. 1024. ISSN: 2041-1723. DOI: 10.1038/s41467-022-28487-2. URL: https://doi.org/10.1038/s41467-022-28487-2.

[45] Yulia Sandamirskaya et al. "Neuromorphic computing hardware and neural architectures for robotics". In: *Science Robotics* 7.67 (2022), eabl8419. DOI: 10.1126/scirobotics.abl8419. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.abl8419. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abl8419.

[46] T. Delbruck and P. Lichtsteiner. "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system". In: *2007 IEEE International Symposium on Circuits and Systems*. 2007, pp. 845–848. DOI: 10.1109/ISCAS.2007.378038.

[47] Thomas Finateu et al. "5.10 A 1280×720 Back-Illuminated Stacked Temporal Contrast Event-Based Vision Sensor with 4.86μm Pixels, 1.066GEPS Readout, Programmable Event-Rate Controller and Compressive Data-Formatting Pipeline". In: *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2020, pp. 112–114. DOI: 10.1109/ISSCC19947.2020.9063149.

[48] Yunjae Suh et al. "A 1280×960 Dynamic Vision Sensor with a 4.95-μm Pixel Pitch and Motion Artifact Minimization". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180436.

[49] Shoushun Chen and Menghan Guo. "Live Demonstration: CeleX-V: A 1M Pixel Multi-Mode Event-Based Sensor". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 1682–1683. DOI: 10.1109/CVPRW.2019.00214.

[50] Theodore Yu et al. "Periodicity detection and localization using spike timing from the AER EAR". In: *2009 IEEE International Symposium on Circuits and Systems*. IEEE. 2009, pp. 109–112.

[51] Shih-Chii Liu et al. "Asynchronous Binaural Spatial Audition Sensor With 2×64×4 Channel Output". In: *IEEE transactions on biomedical circuits and systems* 8.4 (2013), pp. 453–464.

[52] Chiara Bartolozzi et al. "Robots with a sense of touch". In: *Nature Materials* 15.9 (Sept. 2016), pp. 921–925. ISSN: 1476-4660. DOI: 10.1038/nmat4731. URL: https://doi.org/10.1038/nmat4731.

[53] Perla Maiolino et al. "A flexible and robust large scale capacitive tactile system for robots". In: *IEEE Sensors Journal* 13.10 (2013), pp. 3910–3917.

[54] Tom Birkoben et al. "A spiking and adapting tactile sensor for neuromorphic applications". In: *Scientific Reports* 10.1 (Oct. 2020), p. 17260. ISSN: 2045-2322. DOI: 10.1038/s41598-020-74219-1. URL: https://doi.org/10.1038/s41598-020-74219-1.

[55] Jun Chang Yang et al. "Electronic Skin: Recent Progress and Future Prospects for Skin-Attachable Devices for Health Monitoring, Robotics, and Prosthetics". In: *Advanced Materials* 31.48 (2019), p. 1904765. DOI: https://doi.org/10.1002/adma.201904765. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/adma.201904765. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201904765.

[56] Robert W Moncrieff. "An instrument for measuring and classifying odors". In: *Journal of applied physiology* 16.4 (1961), pp. 742–749.

[57] Krishna Persaud and George Dodd. "Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose". In: *Nature* 299.5881 (1982), pp. 352–355.

[58] Chengxiang Wang et al. "Metal Oxide Gas Sensors: Sensitivity and Influencing Factors". In: *Sensors* 10.3 (2010), pp. 2088–2106. ISSN: 1424-8220. DOI: 10.3390/s100302088. URL: https://www.mdpi.com/1424-8220/10/3/2088.

[59] Loic Laplatine et al. "Silicon photonic olfactory sensor based on an array of 64 biofunctionalized Mach-Zehnder interferometers". In: *Opt. Express* 30.19 (Sept. 2022), pp. 33955–33968. DOI: 10.1364/OE.461858. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-30-19-33955.

[60] Nabil Imam and Thomas A. Cleland. "Rapid online learning and robust recall in a neuromorphic olfactory circuit". In: *Nature Machine Intelligence* 2.3 (Mar. 2020), pp. 181–191. ISSN: 2522-5839. DOI: 10.1038/s42256-020-0159-4. URL: https://doi.org/10.1038/s42256-020-0159-4.

[61] Jr John Presper Eckert and John W Mauchly. *Electronic numerical integrator and computer*. US Patent 3,120,606. Feb. 1964.

[62] Gordon E Moore et al. *Cramming more components onto integrated circuits*. 1965.

[63] H. Lee et al. "Sub-5nm All-Around Gate FinFET for Ultimate Scaling". In: *2006 Symposium on VLSI Technology, 2006. Digest of Technical Papers.* 2006, pp. 58–59. DOI: 10.1109/VLSIT.2006.1705215.

[64] TSMC. *TSMC Expands Advanced Technology Leadership with N4P Process*. 2021.

[65] IBM. *IBM Unveils World's First 2 Nanometer Chip Technology, Opening a New Frontier for Semiconductors*. 2021. URL: https://newsroom.ibm.com/2021-05-06-IBM-Unveils-Worlds-First-2-Nanometer-Chip-Technology,-Opening-a-New-Frontier-for-Semiconductors.

[66] Steve Blank. *What the GlobalFoundries' Retreat Really Means*. 2021. URL: https://spectrum.ieee.org/what-globalfoundries-retreat-really-means.

[67] David Schor. *IEDM 2022: Did We Just Witness The Death Of SRAM?* 2022. URL: https://fuse.wikichip.org/news/7343/iedm-2022-did-we-just-witness-the-death-of-sram/.

[68] Stephen W Keckler et al. "GPUs and the future of parallel computing". In: *IEEE micro* 31.5 (2011), pp. 7–17.

[69] Ardavan Pedram et al. "Dark Memory and Accelerator-Rich System Optimization in the Dark Silicon Era". In: *IEEE Design & Test* 34.2 (2017), pp. 39–50. DOI: 10.1109/MDAT.2016.2573586.

[70] Chiara Bartolozzi and Giacomo Indiveri. "Synaptic Dynamics in Analog VLSI". In: *Neural Computation* 19.10 (2007), pp. 2581–2603. DOI: 10.1162/neco.2007.19.10.2581.

[71] L. Chua. "Memristor-The missing circuit element". In: *IEEE Transactions on Circuit Theory* 18.5 (1971), pp. 507–519. DOI: 10.1109/TCT.1971.1083337.

[72] Dmitri B. Strukov et al. "The missing memristor found". In: *Nature* 453.7191 (May 2008), pp. 80–83. DOI: 10.1038/nature06932. URL: https://doi.org/10.1038/nature06932.

[73] Abu Sebastian et al. "Memory devices and applications for in-memory computing". In: *Nature Nanotechnology* 15.7 (June 2020), pp. 529–544. ISSN: 1748-3395. DOI: 10.1038/s41565-020-0655-z. URL: https://doi.org/10.1038/s41565-020-0655-z.

[74] Daniele Ielmini and H.-S. Philip Wong. "In-memory computing with resistive switching devices". In: *Nature Electronics* 1.6 (June 2018), pp. 333–343. ISSN: 2520-1131. DOI: 10.1038/s41928-018-0092-2. URL: https://doi.org/10.1038/s41928-018-0092-2.

[75] Qiming Shao, Zhongrui Wang, and J. Joshua Yang. "Efficient AI with MRAM". In: *Nature Electronics* 5 (2022), pp. 67–68. DOI: 10.1038/s41928-022-00725-x. URL: https://doi.org/10.1038/s41928-022-00725-x.

[76] M. J. Kim et al. "Low power operating bipolar TMO ReRAM for sub 10 nm era". In: *2010 International Electron Devices Meeting*. 2010, pp. 19.3.1–19.3.4. DOI: 10.1109/IEDM.2010.5703391.

[77] Shoichiro Kawashima and Jeffrey S Cross. "FeRAM". In: *Embedded Memories for Nano-Scale VLSIs*. Springer, 2009, pp. 279–328.

[78] L Grenouillet et al. "Performance assessment of BEOL-integrated HfO 2-based ferroelectric capacitors for FeRAM memory arrays". In: *2020 IEEE Silicon Nanoelectronics Workshop (SNW)*. IEEE. 2020, pp. 5–6.

[79] Matthew Jerry et al. "Ferroelectric FET analog synapse for acceleration of deep neural network training". In: *2017 IEEE International Electron Devices Meeting (IEDM)*. 2017, pp. 6.2.1–6.2.4. DOI: 10.1109/IEDM.2017.8268338.

[80] Vincent Garcia and Manuel Bibes. "Ferroelectric tunnel junctions for information storage and processing". In: *Nature communications* 5.1 (2014), pp. 1–12.

[81] D. Ielmini et al. "Physical interpretation, modeling and impact on phase change memory (PCM) reliability of resistance drift due to chalcogenide structural relaxation". In: *2007 IEEE International Electron Devices Meeting*. 2007, pp. 939–942. DOI: 10.1109/IEDM.2007.4419107.

[82] Intel. *Intel Optane Memory - Responsive Memory*. 2015. URL: https://www.intel.com/content/www/us/en/products/details/memory-storage/optane-memory.html.

[83] Panasonic. *8bit Ultra Low Power MN101L*.

[84] F. Arnaud et al. "Truly Innovative 28nm FDSOI Technology for Automotive Micro-Controller Applications embedding 16MB Phase Change Memory". In: *2018 IEEE International Electron Devices Meeting (IEDM)*. 2018, pp. 18.4.1–18.4.4. DOI: 10.1109/IEDM.2018.8614595.

[85] Samsung. *Samsung Electronics Starts Commercial Shipment of eMRAM Product Based on 28nm FD-SOI Process*. 2019. URL: https://news.samsung.com/global/samsung-electronics-starts-commercial-shipment-of-emram-product-based-on-28nm-fd-soi-process.

[86] Feng PAN et al. "Nonvolatile resistive switching memories-characteristics, mechanisms and challenges". In: *Progress in Natural Science: Materials International* 20 (2010), pp. 1–15. ISSN: 1002-0071. DOI: https://doi.org/10.1016/S1002-0071(12)60001-X. URL: https://www.sciencedirect.com/science/article/pii/S100200711260001X.

[87] Cheng-Lin Tsai et al. "Resistive random access memory enabled by carbon nanotube crossbar electrodes". In: *Acs Nano* 7.6 (2013), pp. 5360–5366.

[88] Myoung-Jae Lee et al. "A plasma-treated chalcogenide switch device for stackable scalable 3D nanoscale memory". In: *Nature communications* 4.1 (2013), pp. 1–8.

[89] Enrique A Miranda et al. "Model for the Resistive Switching Effect in Hfo2 MIM Structures Based on the Transmission Properties of Narrow Constrictions". In: *IEEE Electron Device Letters* 31.6 (2010), pp. 609–611.

[90] Hsin-Hung Huang, Wen-Chieh Shih, and Chih-Huang Lai. "Nonpolar resistive switching in the Pt/MgO/Pt nonvolatile memory device". In: *Applied Physics Letters* 96.19 (2010), p. 193505.

[91] Yong-En Syu et al. "Endurance Improvement Technology With Nitrogen Implanted in the Interface of WSiox Resistance Switching Device". In: *IEEE electron device letters* 34.7 (2013), pp. 864–866.

[92] Chao Xie et al. "High-performance nonvolatile Al/AlOx/CdTe: Sb nanowire memory device". In: *Nanotechnology* 24.35 (2013), p. 355203.

[93] Furqan Zahoor, Tun Zainal Azni Zulkifli, and Farooq Ahmad Khanday. "Resistive random access memory (RRAM): an overview of materials, switching mechanism, performance, multilevel cell (MLC) storage, modeling, and applications". In: *Nanoscale research letters* 15.1 (2020), pp. 1–26.

[94] Peng Yao et al. "Fully hardware-implemented memristor convolutional neural network". In: *Nature* 577.7792 (Jan. 2020), pp. 641–646. ISSN: 1476-4687. DOI: 10.1038/s41586-020-1942-4. URL: https://doi.org/10.1038/s41586-020-1942-4.

[95] A. Valentian et al. "Fully Integrated Spiking Neural Network with Analog Neurons and RRAM Synapses". In: *2019 IEEE International Electron Devices Meeting (IEDM)*. 2019, pp. 14.3.1–14.3.4. DOI: 10.1109/IEDM19573.2019.8993431.

[96] R. Khaddam-Aljameh et al. "HERMES Core – A 14nm CMOS and PCM-based In-Memory Compute Core using an array of 300ps/LSB Linearized CCO-based ADCs and local digital processing". In: *2021 Symposium on VLSI Circuits*. 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492362.

[97] Seungchul Jung et al. "A crossbar array of magnetoresistive memory devices for in-memory computing". In: *Nature* 601.7892 (Jan. 2022), pp. 211–216. ISSN: 1476-4687. DOI: 10.1038/s41586-021-04196-6. URL: https://doi.org/10.1038/s41586-021-04196-6.

[98] Weier Wan et al. "A compute-in-memory chip based on resistive random-access memory". In: *Nature* 608.7923 (Aug. 2022), pp. 504–512. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04992-8. URL: https://doi.org/10.1038/s41586-022-04992-8.

[99] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

[100] *Deep Learning's Carbon Emissions Problem*. June 2019. URL: https://www.forbes.com/sites/robtoews/2020/06/17/deep-learnings-climate-change-problem/?sh=1d33d2006b43.

[101] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[102] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575. URL: http://arxiv.org/abs/1409.0575.

[103] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[104] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[105] Vincent Jacob et al. "Spike timing-dependent synaptic depression in the in vivo barrel cortex of the rat". In: *Journal of Neuroscience* 27.6 (2007), pp. 1271–1284.

[106] Manan Suri et al. "Bio-Inspired Stochastic Computing Using Binary CBRAM Synapses". In: *IEEE Transactions on Electron Devices* 60.7 (2013), pp. 2402–2409. DOI: 10.1109/TED.2013.2263000.

[107] Sylvain Saïghi et al. "Plasticity in memristive devices for spiking neural networks". In: *Frontiers in Neuroscience* 9 (2015). ISSN: 1662-453X. DOI: 10.3389/fnins.2015.00051. URL: https://www.frontiersin.org/articles/10.3389/fnins.2015.00051.

[108] Eric Hunsberger and Chris Eliasmith. "Spiking Deep Networks with LIF Neurons". In: *CoRR* abs/1510.08829 (2015). arXiv: 1510.08829. URL: http://arxiv.org/abs/1510.08829.

[109] Iulia M. Comsa et al. "Temporal coding in spiking neural networks with alpha synaptic function". In: *CoRR* abs/1907.13223 (2019). arXiv: 1907.13223. URL: http://arxiv.org/abs/1907.13223.

[110] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. "Training Deep Spiking Neural Networks Using Backpropagation". In: *Frontiers in Neuroscience* 10 (2016). ISSN: 1662-453X. DOI: 10.3389/fnins.2016.00508. URL: https://www.frontiersin.org/articles/10.3389/fnins.2016.00508.

[111] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks". In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 51–63.

[112] Friedemann Zenke and Tim P. Vogels. "The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks". In: *Neural Computation* 33.4 (Mar. 2021), pp. 899–925. ISSN: 0899-7667. DOI: 10.1162/neco_a_01367. eprint: https://direct.mit.edu/neco/article-pdf/33/4/899/1902294/neco_a_01367.pdf. URL: https://doi.org/10.1162/neco%5C_a%5C_01367.

[113] Benjamin Cramer et al. "Surrogate gradients for analog neuromorphic computing". In: *Proceedings of the National Academy of Sciences* 119.4 (2022), e2109194119. DOI: 10.1073/pnas.2109194119. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.2109194119. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2109194119.

[114] Ronald J Williams and David Zipser. "A learning algorithm for continually running fully recurrent neural networks". In: *Neural computation* 1.2 (1989), pp. 270–280.

[115] Guillaume Bellec et al. "A solution to the learning dilemma for recurrent networks of spiking neurons". In: *Nature Communications* 11 (2020). DOI: 10.1038/s41467-020-17236-y.

[116] Benjamin Scellier and Yoshua Bengio. "Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation". In: *Frontiers in Computational Neuroscience* 11 (2017). ISSN: 1662-5188. DOI: 10.3389/fncom.2017.00024. URL: https://www.frontiersin.org/articles/10.3389/fncom.2017.00024.

[117] Axel Laborieux et al. "Scaling Equilibrium Propagation to Deep ConvNets by Drastically Reducing Its Gradient Estimator Bias". In: *Frontiers in Neuroscience* 15 (2021). ISSN: 1662-453X. DOI: 10.3389/fnins.2021.633674. URL: https://www.frontiersin.org/articles/10.3389/fnins.2021.633674.

[118] Axel Laborieux and Friedemann Zenke. *Holomorphic Equilibrium Propagation Computes Exact Gradients Through Finite Size Oscillations*. 2022. DOI: 10.48550/ARXIV.2209.00530. URL: https://arxiv.org/abs/2209.00530.

[119] Geoffrey Hinton. "The Forward-Forward Algorithm: Some Preliminary Investigations". In: (2022).

[120] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *CoRR* abs/1703.03400 (2017). arXiv: 1703.03400. URL: http://arxiv.org/abs/1703.03400.

[121] Alex Nichol, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms". In: *CoRR* abs/1803.02999 (2018). arXiv: 1803.02999. URL: http://arxiv.org/abs/1803.02999.

[122] Axel Laborieux et al. "Synaptic metaplasticity in binarized neural networks". In: *Nature Communications* 12.1 (May 2021), p. 2549. ISSN: 2041-1723. DOI: 10.1038/s41467-021-22768-y. URL: https://doi.org/10.1038/s41467-021-22768-y.

[123] Manu Srinath Halvagal and Friedemann Zenke. "The combination of Hebbian and predictive plasticity learns invariant object representations in deep sensory networks". In: *bioRxiv* (2022). DOI: 10.1101/2022.03.17.484712. eprint: https://www.biorxiv.org/content/early/2022/07/25/2022.03.17.484712.full.pdf. URL: https://www.biorxiv.org/content/early/2022/07/25/2022.03.17.484712.

[124] Elisabetta Chicca and Giacomo Indiveri. "A recipe for creating ideal hybrid memristive-CMOS neuromorphic processing systems". In: *Applied Physics Letters* 116.12 (2020), p. 120501. DOI: 10.1063/1.5142089.

[125] Norman P Jouppi et al. "In-datacenter performance analysis of a Tensor Processing Unit". In: *Proceedings of the 44th annual international symposium on computer architecture*. 2017, pp. 1–12.

[126] Shimeng Yu et al. "Compute-in-memory with emerging nonvolatile-memories: challenges and prospects". In: *2020 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE. 2020, pp. 1–4.

[127] Dovydas Joksas et al. "Committee machines—a universal method to deal with non-idealities in memristor-based neural networks". In: *Nature communications* 11.1 (2020), pp. 1–10.

[128] Mohammed A Zidan, John Paul Strachan, and Wei D Lu. "The future of electronics based on memristive systems". In: *Nature electronics* 1.1 (2018), pp. 22–29.

[129] E. Esmanhotto et al. "High-Density 3D Monolithically Integrated Multiple 1T1R Multi-Level-Cell for Neural Networks". In: *2020 IEEE International Electron Devices Meeting (IEDM)*. 2020, pp. 36.5.1–36.5.4. DOI: 10.1109/IEDM13553.2020.9372019.

[130] Mirko Prezioso et al. "Training and operation of an integrated neuromorphic network based on metal-oxide memristors". In: *Nature* 521.7550 (2015), pp. 61–64.

[131] Stefano Ambrogio et al. "Equivalent-accuracy accelerated neural-network training using analogue memory". In: *Nature* 558.7708 (June 2018), pp. 60–67. DOI: 10.1038/s41586-018-0180-5. URL: https://doi.org/10.1038/s41586-018-0180-5.

[132] Can Li et al. "Long short-term memory networks in memristor crossbar arrays". In: *Nature Machine Intelligence* 1.1 (2019), pp. 49–57.

[133] Zhongrui Wang et al. "In situ training of feed-forward and recurrent convolutional memristor networks". In: *Nature Machine Intelligence* 1.9 (2019), pp. 434–442.

[134] Thomas Dalgaty et al. "In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling". In: *Nature Electronics* 4 (2021), pp. 151–161. DOI: https://doi.org/10.1038/s41928-020-00523-3.

[135] Thomas Dalgaty et al. "Ex-Situ Transfer of Bayesian Neural Networks to Resistive Memory-Based Inference Hardware". In: *Advanced Intelligent Systems* (2021), p. 2000103.

[136] Damien Querlioz et al. "Immunity to device variations in a spiking neural network with memristive nanodevices". In: *IEEE transactions on nanotechnology* 12.3 (2013), pp. 288–295.

[137] Diksha Moolchandani, Anshul Kumar, and Smruti R. Sarangi. "Accelerating CNN Inference on ASICs: A Survey". In: *Journal of Systems Architecture* 113 (2021), p. 101887. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2020.101887. URL: https://www.sciencedirect.com/science/article/pii/S1383762120301612.

[138] Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016).

[139] Saman Parvaneh et al. "Cardiac arrhythmia detection using deep learning: A review". In: *Journal of electrocardiology* 57 (2019), S70–S74.

[140] Paolo Livi and Giacomo Indiveri. "A current-mode conductance-based silicon neuron for address-event neuromorphic systems". In: *2009 IEEE International Symposium on Circuits and Systems*. 2009, pp. 2898–2901. DOI: 10.1109/ISCAS.2009.5118408.

[141] Patrick G Drennan and Colin C McAndrew. "Understanding MOSFET mismatch for analog design". In: *IEEE Journal of solid-state circuits* 38.3 (2003), pp. 450–456.

[142] E. Neftci and G. Indiveri. "A Device Mismatch Compensation Method for VLSI Spiking Neural Networks". In: *Biomedical Circuits and Systems Conference (BioCAS), 2010*. IEEE. 2010, pp. 262–265. DOI: 10.1109/BIOCAS.2010.5709621.

[143] Francesco Maria Puglisi, Paolo Pavan, and Luca Larcher. "Random telegraph noise in HfO<inf>x</inf> Resistive Random Access Memory: From physics to compact modeling". In: *2016 IEEE International Reliability Physics Symposium (IRPS)*. 2016, MY-8-1-MY-8–5. DOI: 10.1109/IRPS.2016.7574624.

[144] Stefano Ambrogio et al. "Noise-Induced Resistance Broadening in Resistive Switching Memory—Part I: Intrinsic Cell Behavior". In: *IEEE Transactions on Electron Devices* 62.11 (2015), pp. 3805–3811. DOI: 10.1109/TED.2015.2475598.

[145] Wolfgang Maass and Henry Markram. "On the computational power of circuits of spiking neurons". In: *Journal of Computer and System Sciences* 69.4 (2004), pp. 593–616. ISSN: 0022-0000. DOI: https://doi.org/10.1016/j.jcss.2004.04.001. URL: https://www.sciencedirect.com/science/article/pii/S0022000004000406.

[146] Mikail Khona and Ila R. Fiete. "Attractor and integrator networks in the brain". In: *Nature Reviews Neuroscience* (2022), pp. 1471–0048. DOI: 10.1038/s41583-022-00642-0. URL: https://doi.org/10.1038/s41583-022-00642-0.

[147] Yann LeCun et al. "Gradient-Based Learning Applied to Document Recognition". In: vol. 86. 11. 1998, pp. 2278–2324.

[148] G.B. Moody and R.G. Mark. "The impact of the MIT-BIH Arrhythmia Database". In: *IEEE Engineering in Medicine and Biology Magazine* 20.3 (2001), pp. 45–50. DOI: 10.1109/51.932724.

[149] Benjamin Cramer et al. "The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[150] George B Moody and Roger G Mark. "The impact of the MIT-BIH arrhythmia database". In: *IEEE Engineering in Medicine and Biology Magazine* 20.3 (2001), pp. 45–50.

[151] F. Corradi, D. Bontrager, and G. Indiveri. "Toward neuromorphic intelligent brain-machine interfaces: An event-based neural recording and processing system". In: *Biomedical Circuits and Systems Conference (BioCAS)*. IEEE. Oct. 2014, pp. 584–587. DOI: 10.1109/BioCAS.2014.6981793.

[152] Nicolas Perez-Nieves et al. "Neural heterogeneity promotes robust learning". In: *bioRxiv* (2021). DOI: 10.1101/2020.12.18.423468. eprint: https://www.biorxiv.org/content/early/2021/03/22/2020.12.18.423468.full.pdf. URL: https://www.biorxiv.org/content/early/2021/03/22/2020.12.18.423468.

[153] E. Chicca et al. "Neuromorphic electronic circuits for building autonomous cognitive systems". In: *Proceedings of the IEEE* 102.9 (Sept. 2014), pp. 1367–1388. ISSN: 0018-9219. DOI: https://doi.org/10.1109/JPROC.2014.2313954.

[154] Thomas Dalgaty, Melika Payvand, Barbara De Salvo, et al. "Hybrid CMOS-RRAM neurons with intrinsic plasticity". In: *IEEE ISCAS*. IEEE. 2019, pp. 1–5.

[155] Chenglong Huang et al. "Efficient and optimized methods for alleviating the impacts of IR-drop and fault in RRAM based neural computing systems". In: *IEEE Journal of the Electron Devices Society* 9 (2021), pp. 645–652.

[156] Tianran Li et al. "A native oxide high-κ gate dielectric for two-dimensional electronics". In: *Nature Electronics* 3.8 (2020), pp. 473–478.

[157] Ruopeng Wang et al. "Recent advances of volatile memristors: Devices, mechanisms, and applications". In: *Advanced Intelligent Systems* 2.9 (2020), p. 2000055.

[158] Alex Fornito, Andrew Zalesky, and Edward T. Bullmore. "Chapter 3 - Connectivity Matrices and Brain Graphs". In: *Fundamentals of Brain Network Analysis*. San Diego: Academic Press, 2016, pp. 89–113. ISBN: 978-0-12-407908-3. DOI: https://doi.org/10.1016/B978-0-12-407908-3.00003-0. URL: https://www.sciencedirect.com/science/article/pii/B9780124079083000030.

[159] William L Hamilton, Rex Ying, and Jure Leskovec. "Representation learning on graphs: Methods and applications". In: *arXiv preprint arXiv:1709.05584* (2017).

[160] Duncan J Watts and Steven H Strogatz. "Collective dynamics of 'small-world'networks". In: *nature* 393.6684 (1998), pp. 440–442.

[161] Ed Bullmore and Olaf Sporns. "Complex brain networks: graph theoretical analysis of structural and functional systems". In: *Nature reviews neuroscience* 10.3 (2009), pp. 186–198.

[162] Alon Loeffler et al. "Topological Properties of Neuromorphic Nanowire Networks". In: *Frontiers in Neuroscience* 14 (2020), p. 184.

[163] Michael Alexander Nugent and Timothy Wesley Molter. "AHaH computing–from metastable switches to attractors to machine learning". In: *PloS one* 9.2 (2014), e85175.

[164] Kelsey S Scharnhorst et al. "Atomic switch networks as complex adaptive systems". In: *Japanese Journal of Applied Physics* 57.3S2 (2018), 03ED02.

[165] Weier Wan et al. "33.1 a 74 TMACS/W CMOS-RRAM neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models". In: *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE. 2020, pp. 498–500.

[166] Gianluca Milano et al. "In materia reservoir computing with a fully memristive architecture based on self-organizing nanowire networks". In: *Nature Materials* (2021), pp. 1–8.

[167] Konstantin K Likharev and Dmitri B Strukov. "CMOL: Devices, circuits, and architectures". In: *Introducing Molecular Electronics*. Springer, 2006, pp. 447–477.

[168] K. Boahen et al. *Address-Event Senders and Receivers: Implementing Direction-Selectivity and Orientation-Tuning*. Ed. by A. Cohen et al. 1998.

[169] Fuxi Cai et al. "Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks". In: *Nature Electronics* 3.7 (2020), pp. 409–418.

[170] Hsiang Tsung Kung and Charles E Leiserson. *Systolic Arrays for VLSI*. Tech. rep. Carnegi-Mellon Univ. Pittsburg PA, 1978.

[171] W. Maass, T. Natschläger, and H. Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations". In: *Neural Computation* 14.11 (2002), pp. 2531–2560.

[172] Ho-Yin Lee et al. "Designing low power of sigma delta modulator for biomedical application". In: *Biomedical Engineering: Applications, Basis and Communications* 17.04 (2005), pp. 181–185.

[173] Federico Corradi and Giacomo Indiveri. "A neuromorphic event-based neural recording system for smart brain-machine-interfaces". In: *IEEE transactions on biomedical circuits and systems* 9.5 (2015), pp. 699–709.

[174] John Moon et al. "Temporal data classification and forecasting using a memristor-based reservoir computing system". In: *Nature Electronics* 2.10 (2019), pp. 480–487.

[175] E. Donati et al. "Discrimination of EMG Signals Using a Neuromorphic Implementation of a Spiking Neural Network". In: *Biomedical Circuits and Systems, IEEE Transactions on* 13.5 (2019), pp. 795–803. DOI: 10.1109/TBCAS.2019.2925454.

[176] Paul J Werbos. "Backpropagation through time: What it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[177] Bojian Yin, Federico Corradi, and Sander M Bohté. "Effective and efficient computation with multiple-timescale spiking recurrent neural networks". In: *International Conference on Neuromorphic Systems 2020*. 2020, pp. 1–8.

[178] P. Merolla et al. "A Multicast Tree Router for Multichip Neuromorphic Systems". In: *Circuits and Systems I: Regular Papers, IEEE Transactions on* 61.3 (Mar. 2014), pp. 820–833. ISSN: 1549-8328. DOI: 10.1109/TCSI.2013.2284184.

[179] E. Painkras et al. "SpiNNaker: A 1-W 18-Core System-on-Chip for Massively-Parallel Neural Network Simulation". In: *IEEE Journal of Solid-State Circuits* 48.8 (Aug. 2013), pp. 1943–1953. ISSN: 0018-9200. DOI: 10.1109/JSSC.2013.2259038.

[180] Ben Varkey Benjamin et al. "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations". In: *Proceedings of the IEEE* 102.5 (2014), pp. 699–716.

[181] K. Likharev and D.Strukov. "CMOL: Devices, circuits, and architectures". In: *Introducing Molecular Electronics* (2005), pp. 447–477.

[182] Jan M Rabaey, Anantha P Chandrakasan, and Borivoje Nikolić. *Digital integrated circuits: a design perspective*. Vol. 7. Pearson education Upper Saddle River, NJ, 2003.

[183] Giacomo Indiveri and Yulia Sandamirskaya. "The Importance of Space and Time for Signal Processing in Neuromorphic Agents: The Challenge of Developing Low-Power, Autonomous Agents That Interact With the Environment". In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 16–28. DOI: 10.1109/MSP.2019.2928376.

[184] Simon J. Thorpe. "Spike arrival times: A highly efficient coding scheme for neural networks". In: *Parallel Processing in Neural Systems and Computers* (Jan. 1990), pp. 91–94.

[185] William B Levy and Victoria G. Calvert. "Communication consumes 35 times more energy than computation in the human cortex, but both costs are needed to predict synapse number". In: *Proceedings of the National Academy of Sciences* 118.18 (2021). ISSN: 0027-8424. DOI: 10.1073/pnas.2008173118. eprint: https://www.pnas.org/content/118/18/e2008173118.full.pdf. URL: https://www.pnas.org/content/118/18/e2008173118.

[186] Thomas Dalgaty et al. "Insect-inspired neuromorphic computing". In: *Current Opinion in Insect Science* 30 (2018). Neuroscience Insect bio-inspired micro and nanotechnologies, pp. 59–66. ISSN: 2214-5745. DOI: https://doi.org/10.1016/j.cois.2018.09.006. URL: https://www.sciencedirect.com/science/article/pii/S2214574518300981.

[187] K. Roy, A. Jaiswal, and P. Panda. "Towards spike-based machine intelligence with neuromorphic computing". In: *Nature* 575 (2019), pp. 607–617. DOI: https://doi.org/10.1038/s41586-019-1677-2.

[188] Johannes Schemmel et al. "Live demonstration: A scaled-down version of the BrainScaleS wafer-scale neuromorphic system". In: *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2012, pp. 702–702. DOI: 10.1109/ISCAS.2012.6272131.

[189] Steve B. Furber et al. "Overview of the SpiNNaker System Architecture". In: *IEEE Transactions on Computers* 62.12 (2013), pp. 2454–2467. DOI: 10.1109/TC.2012.142.

[190] Shih-Chii Liu and Tobi Delbruck. "Neuromorphic sensory systems". In: *Current Opinion in Neurobiology* 20.3 (2010). Sensory systems, pp. 288–295. ISSN: 0959-4388. DOI: https://doi.org/10.1016/j.conb.2010.03.007. URL: https://www.sciencedirect.com/science/article/pii/S0959438810000450.

[191] Thorben Schoepe et al. "Neuromorphic Sensory Integration for Combining Sound Source Localization and Collision Avoidance". In: *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. 2019, pp. 1–4. DOI: 10.1109/BIOCAS.2019.8919202.

[192] Nicoletta Risi et al. "A Spike-Based Neuromorphic Architecture of Stereo Vision". In: *Frontiers in Neurorobotics* 14 (2020), p. 93. ISSN: 1662-5218. DOI: 10.3389/fnbot.2020.568283. URL: https://www.frontiersin.org/article/10.3389/fnbot.2020.568283.

[193] Nicoletta Risi, Enrico Calabrese, and Giacomo Indiveri. "Instantaneous Stereo Depth Estimation of Real-World Stimuli with a Neuromorphic Stereo-Vision Setup". In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401402.

[194] Thomas Dalgaty et al. "Insect-Inspired Elementary Motion Detection Embracing Resistive Memory and Spiking Neural Networks". In: *Biomimetic and Biohybrid Systems*. Vol. 10928. 2018, pp. 115–128. DOI: https://doi.org/10.1007/978-3-319-95972-6_13.

[195] Giulia D'Angelo et al. "Event-Based Eccentric Motion Detection Exploiting Time Difference Encoding". In: *Frontiers in Neuroscience* 14 (2020), p. 451. ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00451. URL: https://www.frontiersin.org/article/10.3389/fnins.2020.00451.

[196] S. Liu V. Chan and A. van Schaik. "AER EAR: A Matched Silicon Cochlea Pair With Address Event Representation Interface". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 54.1 (2007), pp. 48–59. DOI: 10.1109/TCSI.2006.887979.

[197] Angel Jiménez-Fernández et al. "A Binaural Neuromorphic Auditory Sensor for FPGA: A Spike Signal Processing Approach". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.4 (2017), pp. 804–818. DOI: 10.1109/TNNLS.2016.2583223.

[198] N. Imam and T.A. Cleland. "Rapid online learning and robust recall in a neuromorphic olfactory circuit". In: *Nat Mach Intell* 2 (2020), pp. 181–191. DOI: 10.1038/s42256-020-0159-4.

[199] Michele Mastella and Elisabetta Chicca. "A Hardware-Friendly Neuromorphic Spiking Neural Network for Frequency Detection and Fine Texture Decoding". In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401377.

[200] Stefano Ambrogio et al. "Equivalent-accuracy accelerated neural-network training using analogue memory". In: *Nature* 558 (2018), pp. 60–67. DOI: https://doi.org/10.1038/s41586-018-0180-5.

[201] Q. Xia and J.J. Yang. "Memristive crossbar arrays for brain-inspired computing". In: *Nature Material* 18 (2019), pp. 309–323. DOI: https://doi.org/10.1038/s41563-019-0291-x.

[202] Richard J. Przybyla et al. "In-Air Rangefinding With an AlN Piezoelectric Micromachined Ultrasound Transducer". In: *IEEE Sensors Journal* 11.11 (2011), pp. 2690–2697. DOI: 10.1109/JSEN.2011.2157490.

[203] Bruno Fain et al. "Beamforming with AlN-based bimorph piezoelectric micromachined ultrasonic transducers". In: *2021 Smart Systems Integration (SSI)*. 2021, pp. 1–4. DOI: 10.1109/SSI52265.2021.9467016.

[204] P. Muralt et al. "Piezoelectric micromachined ultrasonic transducers based on PZT thin films". In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 52.12 (2005), pp. 2276–2288. DOI: 10.1109/TUFFC.2005.1563270.

[205] Bernard Herrera et al. "AlN PMUT-based Ultrasonic Power Transfer Links for Implantable Electronics". In: *2019 20th International Conference on Solid-State Sensors, Actuators and Microsystems Eurosensors XXXIII (TRANSDUCERS EUROSENSORS XXXIII)*. 2019, pp. 861–864. DOI: 10.1109/TRANSDUCERS.2019.8808320.

[206] Zhou Zhen, Yoshida Shinya, and Tanaka Shuji. "Epitaxial PMnN-PZT/Si MEMS ultrasonic rangefinder with 2m range at 1V drive". In: *Sensors and Actuators A: Physical* 266 (2017), pp. 352–360. ISSN: 0924-4247. DOI: https://doi.org/10.

1016/j.sna.2017.09.058. URL: https://www.sciencedirect.com/science/article/pii/S0924424717312463.

[207] E. I. Knudsen and M. Konishi. "Mechanisms of Sound Localization in the Barn Owl (Tyto alba)". In: *J. Comp. Physiol* 133 (1979), pp. 13–21.

[208] T. Takahashi, A. Bala, and M. Spitzer. "The synthesis and use of the owl's auditory space map". In: *Biological Cybernetics* 89 (2003), pp. 378–387. DOI: https://doi.org/10.1007/s00422-003-0443-5.

[209] C.E. Carr and M. Konishi. "A circuit for detection of interaural time differences in the brain stem of the barn owl". In: *Journal of Neuroscience* 10 (1990), pp. 3227–3246. DOI: https://doi.org/10.1523/JNEUROSCI.10-10-03227.1990.

[210] V. A. Bender et al. "Two Coincidence Detectors for Spike Timing-Dependent Plasticity in Somatosensory Cortex". In: *Journal of Neuroscience* 26 (2006), pp. 4166–4177. DOI: https://doi.org/10.1523/JNEUROSCI.0176-06.2006.

[211] Olivier Caillard, Yehezkel Ben-Ari, and Jean-Luc Gaiarsa. "Long-term potentiation of GABAergic synaptic transmission in neonatal rat hippocampus". In: *Journal of Phisiology* 518 (1999), pp. 109–119. DOI: https://doi.org/10.1111/j.1469-7793.1999.0109r.x.

[212] C.A. Mead, X. Arreguit, and J. Lazzaro. "Analog VLSI model of binaural hearing". In: *IEEE Transactions on Neural Networks* 2 (1991), pp. 230–236. DOI: https://doi.org/10.1109/72.80333.

[213] Das Sarbashis, Dodda Akhil, and Das Saptarshi. "A biomimetic 2D transistor for audiomorphic computing". In: *Nature Communications* 10 (2019), p. 3450. DOI: https://doi.org/10.1038/s41467-019-11381-9.

[214] Thomas Pfeil et al. "Neuromorphic learning towards nano second precision". In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. 2013, pp. 1–5. DOI: 10.1109/IJCNN.2013.6706828.

[215] M. Fabiana Kubke, Dino P. Massoglia, and Catherine E. Carr. "Developmental Changes Underlying the Formation of the Specialized Time Coding Circuits in Barn Owls (Tyto alba)". In: *Journal of Neuroscience* 22.17 (2002), pp. 7671–7679. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.22-17-07671.2002. eprint: https://www.jneurosci.org/content/22/17/7671.full.pdf. URL: https://www.jneurosci.org/content/22/17/7671.

[216] Kazuo Funabiki, Go Ashida, and Masakazu Konishi. "Computation of Interaural Time Difference in the Owl's Coincidence Detector Neurons". In: *Journal of Neuroscience* 31 (2011), pp. 15245–15256. DOI: https://doi.org/10.1523/JNEUROSCI.2127-11.2011.

[217] Lloyd A. Jeffress. "A place theory of sound localization". In: *Journal of Comparative and Physiological Psychology* 41 (1948), pp. 35–39. DOI: https://doi.org/10.1111/j.1469-7793.1999.0109r.x.

[218] Harper Nicol S. and McAlpine David. "Optimal neural population coding of an auditory spatial cue". In: *Nature* 430 (2004), pp. 682–686. DOI: https://doi.org/10.1038/nature02768.

[219] Philip X. Joris, Philip H. Smith, and Tom C. T. Yin. "Coincidence Detection in the Auditory System: 50 Years after Jeffress". In: *Neuron* 21 (1998), pp. 1235–1238.

[220] Ofer Rozen et al. "Monolithic MEMS-CMOS ultrasonic rangefinder based on dual-electrode PMUTs". In: *2016 IEEE 29th International Conference on Micro Electro Mechanical Systems (MEMS)*. 2016, pp. 115–118. DOI: 10.1109/MEMSYS.2016.7421571.

[221] Xiaoyue Jiang et al. "Improving PMUT Transmit Performance via Sub-Micron Thickness Scaling". In: *2018 IEEE International Ultrasonics Symposium (IUS)*. 2018, pp. 1–9. DOI: 10.1109/ULTSYM.2018.8580158.

[222] Richard Przybyla et al. "IN-AIR ULTRASONIC RANGEFINDING AND ANGLE ESTIMATION USING AN ARRAY OF ALN MICROMACHINED TRANSDUCERS". In: *Technical Digest - Solid-State Sensors, Actuators & Microsystems Workshop 2012*. May 2012, pp. 50–53. DOI: 10.31438/trf.hh2012.14.

[223] Minhao Yang et al. "Nanowatt Acoustic Inference Sensing Exploiting Nonlinear Analog Feature Extraction". In: *IEEE Journal of Solid-State Circuits* (2021), pp. 1–1. DOI: 10.1109/JSSC.2021.3076344.

[224] Joontaek Jung et al. "Wafer-level experimental study of residual stress in AlN-based bimorph piezoelectric micromachined ultrasonic transducer". In: *Engineering Research Express* 2 (Oct. 2020). DOI: 10.1088/2631-8695/abc140.

[225] E. Vianello et al. "Resistive Memories for Ultra-Low-Power embedded computing design". In: *2014 IEEE International Electron Devices Meeting*. 2014, pp. 6.3.1–6.3.4. DOI: 10.1109/IEDM.2014.7046995.

[226] A. Grossi et al. "Experimental Investigation of 4-kb RRAM Arrays Programming Conditions Suitable for TCAM". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.12 (2018), pp. 2599–2607. DOI: 10.1109/TVLSI.2018.2805470.

[227] J. Haag et al. "Complementary mechanisms create direction selectivity in the fly". In: *eLife* 5 (2016), pp. 1–15. DOI: https://doi.org/10.7554/eLife.17421.

[228] Syed Ahmed Aamir et al. "An Accelerated LIF Neuronal Network Array for a Large-Scale Mixed-Signal Neuromorphic Architecture". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.12 (2018), pp. 4299–4312. DOI: 10.1109/TCSI.2018.2840718.

[229] Melika Payvand et al. "A neuromorphic systems approach to in-memory computing with non-ideal memristive devices: from mitigation to exploitation". In: *Faraday Discuss* 213 (0 2019), pp. 487–510. DOI: 10.1039/C8FD00114F. URL: http://dx.doi.org/10.1039/C8FD00114F.

[230] Melika Payvand et al. "Analog Weight Updates with Compliance Current Modulation of Binary ReRAMs for On-Chip Learning". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180808.

[231] Melika Payvand and Giacomo Indiveri. "Spike-Based Plasticity Circuits for Always-on On-Line Learning in Neuromorphic Systems". In: *2019 IEEE International*

Symposium on Circuits and Systems (ISCAS). 2019, pp. 1–5. DOI: 10.1109/ISCAS.2019.8702497.

[232] A. Faisal, L Selen, and D Wolpert. "Noise in the nervous system". In: *Nature Reviews Neuroscience* 9.12 (2008), pp. 292–303. DOI: https://doi.org/10.1038/nrn2258.

[233] ST-Microelectronics. *Ultra-low-power Arm Cortex-M4 32-bit MCU+FPU, 100DMIPS, 128KB Flash, 40KB SRAM, analog, AES" STM32L422xx datasheet*. [Online] Accessed: January 2022. 2019. URL: https://www.st.com/resource/en/datasheet/stm32l422kb.pdf.

[234] Daniel Gutierrez-Galan et al. "An Event-Based Digital Time Difference Encoder Model Implementation for Neuromorphic Systems". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021), pp. 1–15. DOI: 10.1109/TNNLS.2021.3108047.

[235] ST-Microelectronics. *Using STM32F4 MCU power modes with best dynamic efficiency*. [Online] Accessed: January 2022. 2019. URL: https://www.st.com/resource/en/application_note/dm00096220-using-stm32f4-mcu-power-modes-with-best-dynamic-efficiency-stmicroelectronics.pdf.

[236] Richard J. Przybyla et al. "3D Ultrasonic Rangefinder on a Chip". In: *IEEE Journal of Solid-State Circuits* 50.1 (2015), pp. 320–334. DOI: 10.1109/JSSC.2014.2364975.

[237] Yihsiang Chiu et al. "A Novel Ultrasonic TOF Ranging System Using AlN Based PMUTs". In: *Micromachines* 12.3 (2021). ISSN: 2072-666X. DOI: 10.3390/mi12030284. URL: https://www.mdpi.com/2072-666X/12/3/284.

[238] Jungdong Jin et al. "Real-Time Sound Localization Using Generalized Cross Correlation Based on 0.13 μm CMOS Proces". In: *Journal of Semiconductor Technology and Science* 14 (2014), pp. 175–183. DOI: 10.1038/s41467-022-29712-8.

[239] Bin Gao et al. "Memristor-based analogue computing for brain-inspired sound localization with in situ training". In: *Nature Communications* 13 (2022). DOI: 10.1038/s41467-022-29712-8. URL: https://doi.org/10.1038/s41467-022-29712-8.

[240] Bernard Widrow and Rodney Winter. "Neural nets for adaptive filtering and adaptive pattern recognition". In: *Computer* 21.3 (1988), pp. 25–39.

[241] Loic Laplatine et al. "Silicon photonic olfactory sensor based on an array of 64 biofunctionalized Mach-Zehnder interferometers". In: *Opt. Express* 30.19 (Sept. 2022), pp. 33955–33968. DOI: 10.1364/OE.461858. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-30-19-33955.

[242] Pierre Maho et al. "Olfactive robot for gas discrimination over several months using a new optoelectronic nose". In: *2019 IEEE International Symposium on Olfaction and Electronic Nose (ISOEN)*. IEEE. 2019, pp. 1–3.

[243] Kiyoshi Asakawa, Yoshimasa Sugimoto, and Shigeru Nakamura. "Silicon photonics for telecom and data-com applications". In: *Opto-Electronic Advances* 3.10 (2020), p. 200011.

[244] Loic Laplatine et al. "System-level integration of active silicon photonic biosensors using Fan-Out Wafer-Level-Packaging for low cost and multiplexed point-of-care

diagnostic testing". In: *Sensors and Actuators B: Chemical* 273 (2018), pp. 1610–1617.

[245] Alphus D Wilson and Manuela Baietto. "Applications and advances in electronic-nose technologies". In: *Sensors* 9.7 (2009), pp. 5099–5148.

[246] Caroline Bushdid et al. "Humans can discriminate more than 1 trillion olfactory stimuli". In: *Science* 343.6177 (2014), pp. 1370–1372.

[247] Enxiao Luan et al. "Silicon photonic biosensors using label-free detection". In: *Sensors* 18.10 (2018), p. 3519.

[248] Giuseppe Antonacci et al. "Ultra-sensitive refractive index gas sensor with functionalized silicon nitride photonic circuits". In: *APL photonics* 5.8 (2020), p. 081301.

[249] R. Halir et al. "Direct and Sensitive Phase Readout for Integrated Waveguide Sensors". In: *IEEE Photonics Journal* 5.4 (2013), pp. 6800906–6800906. DOI: 10.1109/JPHOT.2013.2276747.

[250] P. J. Reyes-Iglesias et al. "High-performance monolithically integrated 120° down-converter with relaxed hardware constraints". In: *Opt. Express* 20.5 (Feb. 2012), pp. 5725–5741. DOI: 10.1364/OE.20.005725. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-20-5-5725.

[251] *NeOse: aryballe's silicon photonics gas sensor*. 2022. URL: https://aryballe.com/our-products/device-solutions/.

[252] Tobias Delbrueck and C Mead. "Bump circuits". In: *Proceedings of International Joint Conference on Neural Networks*. Vol. 1. 1993, pp. 475–479.

[253] Melika Payvand et al. "Error-triggered three-factor learning dynamics for crossbar arrays". In: *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE. 2020, pp. 218–222.

[254] Sourav Dutta et al. "Neural sampling machine with stochastic synapse allows brain-like learning and inference". In: *Nature Communications* 13.1 (2022), pp. 1–10.

[255] Manuel Le Gallo and Abu Sebastian. "An overview of phase-change memory device physics". In: *Journal of Physics D: Applied Physics* 53.21 (2020), p. 213002.

[256] A. Trabelsi et al. "Frequency modulation of conductance level in PCM device for neuromorphic applications". In: *ESSCIRC 2022- IEEE 48th European Solid State Circuits Conference (ESSCIRC)*. 2022, pp. 129–132. DOI: 10.1109/ESSCIRC55480.2022.9911461.

[257] *B1500A Semiconductor Device Parameter Analyzer*. URL: https://www.keysight.com/us/en/assets/7018-03960/technical-overviews/5991-2443.pdf.

[258] Boubacar Traoré et al. "On the origin of low-resistance state retention failure in HfO 2-based RRAM and impact of doping/alloying". In: *IEEE Transactions on Electron Devices* 62.12 (2015), pp. 4029–4036.

[259] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[260] Eduardo Esmanhotto et al. "Experimental Demonstration of Multilevel Resistive Random Access Memory Programming for up to Two Months Stable Neural Networks Inference Accuracy". In: *Advanced Intelligent Systems* 4.11 (2022), p. 2200145.

[261] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[262] Juliane Herpich and Christian Tetzlaff. "Principles underlying the input-dependent formation and organization of memories". In: *Network Neuroscience* 3.2 (2019), pp. 606–634.

[263] Christian Tetzlaff et al. "The use of Hebbian cell assemblies for nonlinear computation". In: *Scientific reports* 5 (2015), p. 12866.

[264] Andreea Lazar, Gordon Pipa, and Jochen Triesch. "SORN: a self-organizing recurrent neural network". In: *Frontiers in computational neuroscience* 3 (2009), p. 23.

[265] Raphaela Kreiser et al. "Error estimation and correction in a spiking neural network for map formation in neuromorphic hardware". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6134–6140.

[266] Ning Qiao et al. "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses". In: *Frontiers in neuroscience* 9 (2015), p. 141. DOI: 10.3389/fnins.2015.00141.

[267] Can Li et al. "Efficient and self-adaptive in-situ learning in multilayer memristor neural networks". In: *Nature communications* 9.1 (2018), pp. 1–8.

[268] Thomas Dalgaty et al. "In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling". In: *Nature Electronics* 4.2 (2021), pp. 151–161.

[269] Fuxi Cai et al. "Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks". In: *Nature Electronics* 3.7 (2020), pp. 409–418.

[270] Rafatul Faria, Kerem Y Camsari, and Supriyo Datta. "Implementing Bayesian networks with embedded stochastic MRAM". In: *AIP Advances* 8.4 (2018), p. 045101.

[271] Abu Sebastian, Manuel Le Gallo, and Evangelos Eleftheriou. "Computational phase-change memory: beyond von Neumann computing". In: *Journal of Physics D: Applied Physics* 52.44 (Aug. 2019), p. 443002. DOI: 10.1088/1361-6463/ab37b6.

[272] Christopher Bengel et al. "Utilizing the Switching Stochasticity of HfO2/TiOx-Based ReRAM Devices and the Concept of Multiple Device Synapses for the Classification of Overlapping and Noisy Patterns". In: *Frontiers in Neuroscience* 15 (2021).

[273] M. Suri et al. "Bio-inspired stochastic computing using binary CBRAM synapses". In: *Electron Devices, IEEE Transactions on* 60.7 (2013), pp. 2402–2409.

[274] Charlotte Frenkel, Jean-Didier Legat, and David Bol. "MorphIC: A 65-nm 738k-Synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning". In: *IEEE transactions on biomedical circuits and systems* 13.5 (2019), pp. 999–1010.

[275] D. Kuzum et al. "Nanoelectronic Programmable Synapses Based on Phase Change Materials for Brain-Inspired Computing". In: *Nano Letters* 12.5 (2012), pp. 2179–

2186. DOI: 10.1021/nl201040y. URL: http://pubs.acs.org/doi/abs/10.1021/nl201040y.

[276] Erika Covi et al. "Analog memristive synapse in spiking networks implementing unsupervised learning". In: *Frontiers in neuroscience* 10.482 (2016), pp. 1–13.

[277] Stefano Ambrogio et al. "Unsupervised learning by spike timing dependent plasticity in phase change memory (PCM) synapses". In: *Frontiers in neuroscience* 10 (2016), pp. 1–12. DOI: 10.3389/fnins.2016.00056.

[278] Melika Payvand and Luke Theogarajan. "From winner-takes-all to winners-share-all: Exploiting the information capacity in temporal codes". In: *Neural computation* 30.3 (2018), pp. 761–791.

[279] Tim P Vogels and LF Abbott. "Gating multiple signals through detailed balance of excitation and inhibition in spiking networks". In: *Nature Neuroscience* 12.4 (2009), p. 483.

[280] C.K. Machens, M.S. Wehr, and A.M. Zador. "Linearity of cortical receptive fields measured with natural sounds". In: *The Journal of neuroscience* 24.5 (2004), p. 1089.

[281] W. Maass, P. Joshi, and E.D. Sontag. "Computational aspects of feedback in neural circuits". In: *PLOS Computational Biology* 3.1 (2007), pp. 1–20.

[282] Siddharth Gaba et al. "Stochastic memristive devices for computing and neuromorphic applications". In: *Nanoscale* 5.13 (2013), pp. 5872–5878.

[283] T. Dalgaty et al. "Hybrid CMOS-RRAM Neurons with Intrinsic Plasticity". In: *International Symposium on Circuits and Systems (ISCAS), 2019*. IEEE. 2019. DOI: 10.1109/ISCAS.2019.8702603.

[284] Alessandro Grossi et al. "Fundamental variability limits of filament-based RRAM". In: *2016 IEEE International Electron Devices Meeting (IEDM)*. IEEE. 2016, pp. 4–7.

[285] Joseph M Brader, Walter Senn, and Stefano Fusi. "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics". In: *Neural Computation* 19.11 (2007), pp. 2881–2912.

[286] Herbert Jaeger and Harald Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication". In: *Science* 304.5667 (2004), pp. 78–80.

[287] JSP Giraldo and Marian Verhelst. "Hardware acceleration for embedded keyword spotting: Tutorial and survey". In: *ACM Transactions on Embedded Computing Systems (TECS)* 20.6 (2021), pp. 1–25.

[288] G.B. Ermentrout and D.H. Terman. *Mathematical Foundations of Neuroscience*. Interdisciplinary Applied Mathematics. Springer New York, 2010. ISBN: 9780387877082. URL: https://books.google.fr/books?id=0fLdzaFgtjcC.

[289] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[290] KyungHyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259 (2014). arXiv: 1409.1259. URL: http://arxiv.org/abs/1409.1259.

[291] C Koch and A Zador. "The function of dendritic spines: devices subserving biochemical rather than electrical compartmentalization". In: *Journal of Neuroscience* 13.2 (1993), pp. 413–422. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.13-02-00413.1993. eprint: https://www.jneurosci.org/content/13/2/413.full.pdf. URL: https://www.jneurosci.org/content/13/2/413.

[292] B. Alberts et al. *Essential cell biology (4th edition)*. Garland publishing, 2013.

[293] Karen Runge, Carlos Cardoso, and Antoine de Chevigny. "Dendritic Spine Plasticity: Function and Mechanisms". In: *Frontiers in Synaptic Neuroscience* 12 (2020). ISSN: 1663-3563. DOI: 10.3389/fnsyn.2020.00036. URL: https://www.frontiersin.org/articles/10.3389/fnsyn.2020.00036.

[294] Menahem Segal. "Dendritic spines and long-term plasticity". In: *Nature Reviews Neuroscience* 6 (2005). ISSN: 1471-0048. DOI: 10.1038/nrn1649. URL: https://doi.org/10.1038/nrn.2018.16.

[295] Miquel Bosch and Yasunori Hayashi. "Structural plasticity of dendritic spines". In: *Current Opinion in Neurobiology* 22.3 (2012). Synaptic structure and function, pp. 383–388. ISSN: 0959-4388. DOI: https://doi.org/10.1016/j.conb.2011.09.002. URL: https://www.sciencedirect.com/science/article/pii/S0959438811001462.

[296] Marc P. Forrest, Euan Parnell, and Peter Penzes. "Dendritic structural plasticity and neuropsychiatric disease". In: *Nature Reviews Neuroscience* 19 (2018). ISSN: 1471-0048. DOI: 10.1038/nrn.2018.16. URL: https://doi.org/10.1038/nrn.2018.16.

[297] Henry Clavering Tuckwell. *Introduction to theoretical neurobiology: linear cable theory and dendritic structure*. Vol. 1. Cambridge University Press, 1988.

[298] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[299] Hojoon Ryu et al. In: *Scientific Reports* 9.1 (2019), pp. 255–258. DOI: 10.1038/s41598-019-56816-x. URL: https://doi.org/10.1038/s41598-019-56816-x.

[300] Erika Covi et al. "Ferroelectric Tunneling Junctions for Edge Computing". In: *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2021, pp. 1–5. DOI: 10.1109/ISCAS51556.2021.9401800.

[301] Vincent Garcia and Manuel Bibes. In: *Nature Communications* 5.1 (2014). DOI: 10.1038/ncomms5289. URL: https://doi.org/10.1038/ncomms5289.

[302] R. Fontanini et al. "Polarization switching and interface charges in BEOL compatible Ferroelectric Tunnel Junctions". In: *ESSDERC 2021 - IEEE 51st European Solid-State Device Research Conference (ESSDERC)*. 2021, pp. 255–258. DOI: 10.1109/ESSDERC53440.2021.9631812.

[303] Sumit Bam Shrestha and Garrick Orchard. "SLAYER: Spike Layer Error Reassignment in Time". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 1419–1428.

[304] Malu Zhang et al. "Supervised learning in spiking neural networks with synaptic delay-weight plasticity". In: *Neurocomputing* 409 (2020), pp. 103–118.

[305] Ian J Goodfellow et al. "An empirical investigation of catastrophic forgetting in gradient-based neural networks". In: *arXiv preprint arXiv:1312.6211* (2013).

[306] Arnon Amir et al. "A Low Power, Fully Event-Based Gesture Recognition System". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 7388–7397. DOI: 10.1109/CVPR.2017.781.

[307] SR Nandakumar et al. "Mixed-precision deep learning based on computational memory". In: *Frontiers in Neuroscience* 14 (2020), p. 406.