Conception de systèmes efficaces en énergie dédiés à l'inférence bayésienne exploitant des nouvelles technologies mémoires

Clément Turck

THESE DE DOCTORAT

NNT : 2023UPAST166

Conception de systèmes efficaces en énergie dédiés à l'inférence bayésienne exploitant des nouvelles technologies mémoires

Energy efficient bayesian reasoning using novel memory technologies

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 575, electrical, optical, bio : physics and engineering (EOBE)
Spécialité de doctorat : Electronique, Photonique et Micro-Nanotechnologies
Graduate School : Sciences de l'ingénierie et des systèmes Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche Centre de Nanosciences et de Nanotechnologies (Université Paris-Saclay, CNRS), sous la direction de **Damien QUERLIOZ**, Directeur de Recherche, le co-encadrement de **Jean-Michel PORTAL**, Professeur des Universités

Thèse soutenue à Paris-Saclay, le 27 novembre 2023, par

Clément TURCK

Composition du Jury

Membres du jury avec voix délibérative

Jérôme SAINT MARTIN Professeur des universités, Ecole Normale Supérieure Paris-Saclay	Président
Alberto BOSIO Professeur des universités, Ecole Centrale de Lyon	Rapporteur & Examinateur
Louis HUTIN Cadre scientifique des EPIC (HDR), CEA, Université Grenoble Alpes	Rapporteur & Examinateur
Marina REYBOZ Cadre scientifique des EPIC, CEA, Université Grenoble Alpes	Examinatrice

université PARIS-SACLAY
ÉCOLE DOCTORALE
Physique et ingénierie : Electrons, Photons, Sciences du vivant (EOBE)

Titre : Conception de systèmes efficaces en énergie dédiés à l'inférence bayésienne exploitant des nouvelles technologies mémoires

Mots clés : Inférence bayésienne, Mémoire, Efficacité énergétique

Résumé : L'intelligence artificielle (IA) est centrale aux avancées actuelles, dont l'analyse de données et la médecine. Des modèles comme ChatGPT et Dall-E exploitent le cloud, centralisant d'immenses volumes de données. Cependant, leur consommation énergétique est problématique, spécialement pour des appareils à ressources limitées, vus comme une alternative à la centralisation. Par exemple, l'entraînement du modèle Llama-2 de Meta consomme autant qu'annuellement 33 foyers français. Une solution serait d'entraîner le modèle hors de l'appareil, puis d'exécuter l'inférence sur celui-ci, malgré les contraintes de performance. Au-delà des défis énergétiques, l'IA est souvent pointée du doigt pour son manque de transparence. En médecine ou dans la conduite autonome, il est impératif de comprendre le raisonnement derrière chaque décision prise par l'IA.

Dans cette thèse, nous nous concentrons sur l'inférence bayésienne, une solution possible pour des modèles IA plus économes en énergie, explicites et transparents.

Dans le premier chapitre, nous explorons l'architecture Von-Neumann des circuits actuels puis nous abordons l'avancée des nouvelles architectures inspirées du fonctionnement du cerveau, mettant l'accent sur les memristors et leur potentiel pour le calcul en mémoire. Dans le deuxième et troisième chapitre, nous présentons une « machine bayésienne », de sa théorie à sa conception. L'explication de son fonctionnement est abordée, jusqu'à une application réelle de reconnaissance de geste, en passant par l'architecture de cette machine bayésienne. Dans cette partie, nous nous penchons sur une méthode de calcul qui permet de diminuer la consommation énergétique : le calcul stochastique. Elle est présentée comme une solution réduisant également l'encombrement des multiplicateurs nécessaires à l'inférence bayésienne.

Les applications pratiques de cette approche sont ensuite examinées dans la conception des circuits et les possibilités offertes pour une application pratique, la reconnaissance des gestes, sont explorées. Ensuite, la performance de l'inférence bayésienne pour cette application est évaluée dans des contextes à faible volume de données, et le caractère explicable de la machine est étudié. Les étapes du processus de fabrication sont ensuite détaillées, en mettant en lumière les innovations récentes dans la structuration et la programmation de cette machine. Nous explorons également les défis et les opportunités associés à la mesure et à la caractérisation de ces systèmes, avant de conclure avec une discussion sur les perspectives d'amélioration de l'efficacité énergétique. Le quatrième chapitre est consacré à une comparaison entre deux architectures de calcul pour une machine, le calcul stochastique étudié précédemment et le calcul logarithmique nouvellement développé. Nous faisons la comparaison tant en termes d'application, sur la reconnaissance de geste et également sur une application plus médicale qu'est la reconnaissance des cycles du sommeil, qu'en termes énergétiques. Par la suite, une intégration de cette puce dans un système est présentée, composé d'un cœur RISC-V et d'une nouvelle puce multimode de calcul avec une plus grande capacité mémoire. En conclusion, la recherche, tant du côté software, avec les différentes applications possibles de l'inférence bayésienne, que du côté hardware, avec l'intégration des nouvelles technologies mémoires comme les memristors et l'intégration des systèmes de collecte d'énergie, est présentée comme une étape importante dans l'innovation de l'intelligence artificielle à faible consommation d'énergie et plus particulièrement de l'intelligence artificielle embarquée.

Title : Energy efficient bayesian reasoning using novel memory technologies

Keywords : Energy efficient, Bayesian Inference, Memory

Abstract : Artificial Intelligence (AI) plays a key role in modern advancements, especially in data analysis and healthcare. Large models, such as ChatGPT and Dall-E, rely on cloud platforms, consolidating vast amounts of data. However, their high energy consumption poses challenges, particularly for devices with limited resources, which are seen as a decentralizing alternative. For context, training the Llama-2 model from Meta uses as much energy as 33 French households consume in a year. One solution might be to train the model externally and then run the inference on the device itself, even with performance challenges. Beyond energy concerns, AI's lack of transparency is a critique. In medical applications or autonomous driving, understanding AI decision-making is crucial.

In this thesis, we will investigate Bayesian inference, a potential answer for energy efficient, clearer, and more transparent AI models.

In the first chapter, we delve into the von Neumann architecture of current circuits, followed by a look at the advancements in new architectures inspired by the brain's function. The role of memristors and their potential for in-memory computing are emphasized. In the second and third chapters, we discuss a "Bayesian machine", from its theoretical foundation to its design. This includes an explanation of its operation and its real-world application in gesture recognition, taking into account the architecture of this Bayesian machine. The machine relies on stochastic computing, a computing method that reduces energy consumption. This approach also helps minimize the size of multipliers used in Bayesian inference. Its practical applications in circuit design are then assessed, particularly for gesture recognition. Additionally, the efficiency of Bayesian inference in scenarios with limited data is evaluated and the machine's explainability is explored. The next chapter details the manufacturing process of the machine, spotlighting recent innovations in the structuring and programming of the machine. Challenges and opportunities associated with measuring and characterizing these systems are also delved into, concluding with thoughts on energy efficiency improvements. The fourth chapter contrasts two computational architectures for machines: the previously discussed stochastic computing and the newly introduced logarithmic computing. They are compared both in terms of applications - from gesture recognition to a medical application of sleep cycle detection - and in energy efficiency. The integration of the chip into a system with a RISC-V core and a new multimode computing chip with enhanced memory capacity is then presented. In conclusion, research, both from the software perspective with the varied applications of Bayesian inference, and the hardware side with the adoption of innovative memory technologies like memristors and energy harvesting systems, signifies a crucial step in the advancement of low-energy AI, specifically embedded AI.

À ma grand-mère qui, je sais, d'où elle est, est fière de moi.
À ma famille.

Remerciements

Je tiens avant tout à exprimer ma profonde gratitude à mon directeur de thèse, Damien pour m'avoir déjà supporté puis soutenu et encouragé pendant ces trois et quelques années. Ton expertise et tes conseils précieux m'ont permis de réaliser pleinement ce travail de recherche. J'ai vraiment passé trois belles années à tes côtés. Je remercie également Jean-Michel qui m'a co-encadré. Nos échanges et tes conseils m'ont énormément apportés. Je tiens à remercier l'ensemble des membres du jury, Louis Hutin, Alberto Bosio, Marina Reyboz, Jérome Saint-Martin et David Novo pour d'une part avoir bien voulu faire partie de mon jury et d'avoir étudié mon manuscrit et d'autre part pour les critiques constructives et leurs discussions enrichissantes.

J'ai fait partie d'une équipe formidable durant cette thèse, IntegNano. Tout d'abord, merci à mon binôme, Kamel, sans qui aucune des puces n'auraient pu être envoyées en production, nous avons été vraiment complémentaires sur l'intégralité de nos travaux respectifs. Nous avons tellement travaillés ensemble que nous sommes co-auteurs de tous nos articles. Je te souhaite de trouver ton équipe de Foot miniature. Merci à Marie, jumelle de thèse comme elle aimait m'appeler, nous avons commencé le même jour et fini avec juste une semaine d'écart. Notre soutient mutuel lors de la rédaction m'a permis d'avancer rapidement. Je te souhaite une belle réussite dans ta carrière pour que tu puisse réaliser ton rêve d'achat. Merci à Maryam, ma co-bureau pendant un an et demi, j'ai vraiment aimé nos discussions tant sur le travail que sur la vie en général. Je te souhaite une belle réussite dans ta carrière. Merci à Atreya, ton enthousiasme et ta passion pour la nourriture m'a toujours fasciné. Merci à Tifenn pour avoir lancé le projet avec Kamel. Merci à Théo pour tes discussions sur l'architecture des ordinateurs et le RISC-V, j'ai beaucoup appris. Merci à Adrien R., l'éternel stagiaire, en stage dans l'équipe depuis la licence, tu as su réaliser le travail d'un doctorant pour la mise en place des setups de test. J'ai vraiment adoré travailler avec toi. Merci à Adrien P. je te souhaite de trouver ton expression dans le théâtre. Merci à Thomas pour ta bonne humeur dans le labo. Merci à Tanvi pour tes conseils sur les enfants, je te souhaite beaucoup de bonheur avec ta famille. Merci à Gyan pour ton entrain, ta bonne humeur et ta joie de vivre. Merci à Akib pour tes anecdotes intéressantes sur ton pays. Merci à Liza, j'adore vraiment tes tableaux. Merci au reste de l'équipe, Djohan qui va bientôt finir, Bastien, Guillaume. Enfin, merci à toutes les personnes avec qui j'ai travaillé, discuté et partagé des bons moments, Thibaut, Rohit, Fadi, Elmer, Maïkane, Matthieu, Yanis, Song et Louis.

Je remercie également toute ma famille et belle famille, c'est grâce à eux que j'ai pu terminer toutes ces longues années d'études. Je me souviens des séances de révisions avec ma maman le soir pour apprendre mes leçons, mes poésies, mes tables de multiplications. Je me souviens des séances de bricolage avec mon papa, enfin, j'étais plus le commis qui tenait la lampe mais c'est des moments précieux pour moi. Je me souviens des séances de jeux avec mon frère, de nos chamailleries aussi. Je me souviens des moments passés avec mes grand-parents, des sorties

au parc avec mamie Roselyne et tonton Nono, des nuits passées chez mamie Françoise et mon parrain, à jouer aux petites voitures. Je me souviens des vacances passées chez ma marraine à regarder Fort-Boyard sur le petit écran de la télé, des sorties avec mes cousines et leurs enfants. Je me souviens des anniversaires chez Nathalie, avec Benjamin et Laurine que je considère comme mes cousins. Je me souviens et, là, c'est plus récent (enfin 10 ans quand même) de l'accueil chaleureux, de l'amour immédiat de Frédéric et Isabelle que j'ai eu en arrivant dans la famille de ma femme. Tous ces moments, tous ces souvenirs m'ont vraiment aidé dans la réalisation de cette thèse et je veux vraiment dire un grand **merci** à tout le monde pour leur soutien.

Guillaume, je ne vais pas écrire un paragraphe sur toi uniquement mais tu as ta petite marque dans cette thèse. Je sais que j'ai fait un sacrilège en inversant les titres mais après toutes ces années, tu me le pardonneras. Merci, d'être mon meilleur ami depuis l'école primaire, merci d'avoir été toujours présent à mes côtés. Un grand merci aussi à mes amis de longue date pour les bons moments passés ensemble, Thibaut, Fouad, Son-Dorian, Quentin, Julien, Rémy et Antoine.

Enfin, la dernière personne mais la plus importante à mes yeux, celle qui m'a soutenu quand j'avais des coups de mou, celle qui m'a encouragé, celle qui m'a poussé à aller jusqu'au bout, celle qui m'a donné un fils, mon petit Léandre qui a assisté dans le ventre de sa maman à ma soutenance, ma chipoustouquette chérie comme je l'appelle souvent, bref, Alicia, ma femme, je t'aime et merci.

Voilà, j'ai fini je pense. Je m'excuse si j'ai oublié quelqu'un, je remercie toutes celles avec qui j'ai discuté et qui m'ont écouté.

Normalement, les remerciements dans les films sont à la fin, mais pour cette thèse, je vais faire une exception. Je vous laisse maintenant profiter de la lecture et bon visionnage.

Clément

Le petit post-scriptum de la fin : des parties de ce manuscrit ont été révisées avec l'aide d'un grand modèle de langage (OpenAI ChatGPT).

Contents

Introduction **1**

1 Énergie et Mémoires résistives : Un voyage inattendu **7**
1.1 Vers une optimisation de l'efficacité énergétique des circuits... 8
1.1.1 Évolution des circuits et de l'architecture de Von-Neumann 8
1.1.2 Architecture inspirée par le Cerveau 13
1.2 ... A une nouvelle technologie de mémoire : les memristors 17
1.2.1 Mémoires actuelles : SRAM, DRAM, Flash 17
1.2.2 Différents types de memristors 19
1.2.3 Leur intégration pour le calcul en mémoire 21

2 La communauté de la Machine Bayésienne **29**
2.1 Principes Fondamentaux de l'Approche Bayésienne 30
2.1.1 Indépendance Conditionnelle : Au-delà de l'Approximation Naïve 31
2.1.2 Techniques de Multiplication dans l'Analyse Bayésienne 32
2.2 Intégration de la Machine Bayésienne dans la Conception des Circuits 34
2.2.1 Exploration de l'Architecture 34
2.2.2 L'Importance du Choix des Seeds 38
2.3 Application Pratique : Reconnaissance des Gestes 41
2.3.1 Fonctionnement de l'Application 41
2.3.2 Analyse des Résultats 43
2.3.3 Vers une Explicabilité Accrue 49
2.3.4 Performances sur des Ensembles de Données Réduits 53

3 Machine Bayésienne II, le retour du roi **57**
3.1 Design et Fabrication de la Puce Bayésienne 58
3.1.1 Processus de Fabrication 58
3.1.2 Structure 2T2R 59
3.1.3 Étape de Programmation 59
3.1.4 Étapes de Lecture et d'Inférence 61
3.1.5 Synthèse : Étapes pour l'Utilisation de la Machine Bayésienne 63

3.2 Analyse Approfondie : Mesures et Caractérisation 70
3.2.1 Exploration des Applications à Faible Tension : La Puce sans Package . . . 70
3.2.2 La Puce Packagée : Comprendre les Perturbations de Lecture 72
3.3 Estimations des Performances Énergétiques 76
3.3.1 Simulations : Un Outil pour l'Analyse Énergétique 76
3.3.2 Conclusion et Perspectives d'Améliorations 79

4 Machine Bayésienne III, les deux modes! 83
4.1 Une Machine Bayésienne Logarithmique 84
4.1.1 Le Calcul Logarithmique 84
4.1.2 Une Nouvelle Application : les Cycles du Sommeil 85
4.1.3 Comparaison des Deux Méthodes de Calcul 89
4.2 Mesures de la Nouvelle Puce 94
4.2.1 Reconnaissance des cycles du sommeil 94
4.2.2 Intégration de la Machine Bayésienne dans un Coeur RISC-V 95
4.3 Vers une Version Agrandie et Multi-Modes de la Machine Bayésienne 97

Conclusions et Perspectives 101

List of publications 107

A Annexe 1 : A Multimode Hybrid Memristor-CMOS Prototyping Platform Supporting Digital and Analog Projects 109

B Annexe 2 : Powering AI at the Edge: A Robust, Memristor-based Binarized Neural Network with Near-Memory Computing and Miniaturized Solar Cell 113

C Annexe 3 : In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling 129

Bibliography 150

List of Figures

1 **Image de moi présentant ma thèse, générée par Stable Diffusion.** Entraînement du modèle en utilisant des photos de moi et l'environnement kohya_ss (https://github.com/bmaltais/kohya_ss) puis générée en utilsant l'environnement AUTOMATIC1111 (https://github.com/AUTOMATIC1111/stable-diffusion-webui) avec le modèle SD-XL [1] en 60 itérations. 3

1.1 **Évolution du transistor : du premier aux 1000 milliards sur puce. a** Le premier transistor, développé par Walter Brattain et John Bardeen en 1947, source:Alcatel-Lucent USA, Inc **b** Photo du die du i9-12900K de Intel, source:Intel et **c** Évolution de la technologie de fabrication d'Intel, du passé au futur, source:https://www.intel.com/content/www law.html 9

1.2 **Évolution du MOSFET. a** Évolution de l'architecture du MOSFET, source : Samsung, https://www.samsungfoundry.com/foundry/homepage.do **b** Graphe illustrant la fin de l'augmentation exponentielle des performances; la loi de Moore va sûrement se terminer aussi, tiré de [2] 10

1.3 **Architecture des ordinateurs actuels. a** Schéma d'une architecture d'un CPU simplifiée et **b** d'un GPU, adapté de la documentation CUDA de NVIDIA 11

1.4 **Limites de l'architecture de Von Neumann. a** Schéma d'une architecture d'un CPU simplifiée avec le goulot d'étranglement de Von Neumann à cause de la séparation physique de la mémoire et du calcul et **b** Coûts énergétiques pour différentes opérations arithmétique comparés aux coûts pour l'accès mémoire, tiré de [3]. L'accès à la DRAM est 4 ordres de grandeur plus énergivore que l'addition sur 8 bits 13

1.5 **Schéma d'un neurone.** Les signaux neuronaux entrent par les dendrites, des branches vastes et épaisses qui émergent du soma, et sont recouvertes de synapses. Ces signaux, appelés potentiels d'action, transitent ensuite vers d'autres neurones via des axones. Ces derniers, capables de s'étendre sur de grandes distances, peuvent atteindre plus d'un mètre de long et se ramifient de nombreuses fois pour permettre une communication efficace. Un seul neurone peut ainsi intégrer plus de 10 000 synapses. (Schéma adapté de Shuai Li) 14

1.6 **Applications de l'inspiration du cerveau. a** Réseau de neurones pour identifier un chat et **b** Architecture pour le calcul proche mémoire où les neurones (unités de calcul) sont co-localisés avec les synapses (mémoire) 17

1.7 **Différentes technologies de mémoire. a** Mémoire SRAM composée de 6 transistors formant deux inverseurs tête-bêche **b** Mémoire DRAM comprenant un condensateur C réalisant le stockage, connecté en série avec un transistor FET **c** Mémoire FLASH, le stockage est ici effectué par une porte flottante d'un transistor FET et peut être utilisé pour **d** la structure NOR ou **e** la structure NAND, tiré de [4] 19

1.8 **Hiérarchie de la mémoire et comblement du trou avec les nouvelles technologies de mémoires** 20

1.9 **Structures et récapitulatif des nouvelles technologies mémoire** 22

1.10 **Structures d'utilisation des nouvelles technologies mémoire. a** 1T1R, cellule composée d'un transistor et d'un memristor, **b** cellule composée uniquement d'un memristor, l'application d'une différence de tension entre BL et WL sur la cellule verte (celle programmée) se répercute sur l'ensemble des autres memristors, en rouge, et **c** cellule composée d'un sélectionneur et d'un memristor (1S1R), seuls les cellules en rouge reçoivent la différence de tension, celles en bleu sont intactes. 23

1.11 **Architecture crossbar pour le calcul en mémoire analogique a** Schéma de l'architecture crossbar pour réaliser les opérations MAC d'un **b** réseau de neurones, **c** structure croisée avec composée de RRAM sur une seule couche et **d** sur plusieurs couches (tiré de [5] et [6]) 25

1.12 **Architecture crossbar pour le calcul en mémoire avec sommation des résistances a b c d** (tiré de [6]) 26

2 La communauté de la Machine Bayésienne **29**

2.1 **Calcul stochastique pour deux flux de bits. a** opération de multiplication si les flux ne sont pas corrélés, le résultat est bien le produit des deux flux en entrée et **b** si les flux sont corrélés, le résultat ne correspond pas au produit des deux flux en entrée 33

2.2 **Architecture générale de la machine bayésienne. a** Architecture générale. Les vraisemblances sont stockées dans des matrices de mémoire de vraisemblance implémentées par des matrices de memristors. Les observations du monde réel choisissent la valeur de probabilité appropriée à partir des matrices de mémoire de vraisemblance, sur la base desquelles des flux de bits stochastiques proportionnels sont générés et multipliés par un multiplicateur stochastique. En sortie, les flux de bits encodent naturellement la distribution a posteriori. **b** Architecture de la machine bayésienne avec des observations non conditionnellement indépendantes. Cette architecture réalise une inférence bayésienne non naïve suivant l'équation 2.5, en regroupant les observations O_2 et O_3 dans la même colonne. **c** Optimisation de la machine bayésienne pour le matériel. Des nombres aléatoires (RND) sont générés à l'aide de registres à décalage à rétroaction linéaire (LFSRs), partagés par colonne, puis convertis en utilisant des circuits numériques "Gupta" en une série de bits aléatoires proportionnels à la probabilité appropriée. De plus, les vraisemblances sont normalisées par la valeur de vraisemblance maximale de la colonne afin de maximiser la vitesse de convergence de la machine. La multiplication stochastique est mise en œuvre à l'aide d'une porte logique ET à un seul bit. 36

2.3 **Schéma du circuit comparateur de Gupta** [7], utilisé pour générer le flux de bits aléatoires de vraisemblance en comparant la probabilité lue en mémoire avec le nombre aléatoire généré par le LFSR. Pour des raisons de compacité, le circuit est présenté ici dans une version à quatre bits. Une version à huit bits est mise en œuvre sur la puce. 37

2.4 **Corrélations des nombres aléatoires générés par quatre LFSR, avec des graines initialement choisies au hasard** Chaque graphique présente la sortie de l'un des quatre LFSR de la machine bayésienne, en fonction de la sortie d'un autre LFSR. Chaque graphique contient 255 points correspondant aux 255 cycles de fonctionnement de la machine bayésienne. Les graphiques sur la diagonale (LFSR1/LFSR1, LFSR2/LFSR2, LFSR3/LFSR3, LFSR4/LFSR4) apparaissent comme des lignes x=y, par définition. La présence de motifs très discernables dans certains des graphiques (LFSR1/LFSR3, LFSR1/LFSR4) indique l'existence d'une forte corrélation entre la sortie de certains LFSR. D'un autre côté, les sorties de certains LFSR semblent largement non corrélées (LFSR1/LFSR2, LFSR2/LFSR3, LFSR2/LFSR4). Les graines pour les quatre LFSR sont, respectivement, en représentation hexadécimale : 50, E9, 10, et C6. 39

2.5 **Corrélations des nombres aléatoires générés par quatre LFSR, avec les graines optimales.** Cette Figure, tracée avec les mêmes conventions que la Fig.2.4, montre l'absence de corrélation évidente entre les sorties des différents LFSR. Les graines pour les quatre LFSR sont, respectivement, en représentation hexadécimale : EB, FB, 7F, et 5C. 40

2.6 **Résultats de la machine bayésienne à une tâche pratique de reconnaissance de gestes.** *Précision moyenne* en fonction du nombre de cycles pour deux types de calcul : en utilisant une méthode économe en énergie en ne prenant en compte que le premier '1' pour la décision (en rouge) et en utilisant le calcul stochastique conventionnel en utilisant le nombre maximum de '1' pour la décision (en bleu). Les ombres autour du graphique montrent un écart-type de la précision moyenne sur les dix sujets. 44

2.7 **Impact des erreurs sur les bits de mémoire sur la précision de la tâche de reconnaissance des gestes**. Précision moyenne simulée sur la tâche de reconnaissance des gestes, lorsque des erreurs de bits ont été artificiellement introduites dans les bits de mémoire, en fonction du taux d'erreur de bit de mémoire. La simulation a été répétée 40 fois, les ombres autour du graphique montrent l'écart type de la précision moyenne le long de ces répétitions. Les résultats sont présentés pour le mode power-conscious avec un seuil à 50, 100, et 255 cycles, et pour le mode de calcul stochastique conventionnel utilisant tous les 255 cycles. 45

2.8 **Impact des erreurs uniques sur la précision de la tâche de reconnaissance des gestes**. Précision moyenne simulée sur la tâche de reconnaissance des gestes, lorsque des erreurs uniques ont été artificiellement introduites dans le calcul, en fonction du taux d'erreur de bit de mémoire. Les erreurs uniques sont introduites à la sortie des circuits Gupta (c'est-à-dire sur les flux de bits stochastiques créés par chaque bloc de vraisemblance). La simulation a été répétée 100 fois, les ombres autour du graphique montrent un écart type de la précision moyenne le long de ces répétitions. Les résultats sont présentés pour le mode power-conscious avec un seuil à 50, 100, et 255 cycles, et pour le mode de calcul stochastique conventionnel utilisant tous les 255 cycles. 47

2.9 **Capacité de la machine bayésienne à identifier les situations incertaines. a** Illustration schématique des deux signatures d'une machine bayésienne incertaine. **b** Réponse moyenne simulée de la machine bayésienne, pour la tâche de reconnaissance de gestes, dans des situations où un geste présenté à la machine bayésienne peut provenir d'un sujet différent de celui sur lequel la machine a été entraînée. Abscisse : geste présenté. Pour chaque sujet, les quatre points représentent les gestes un, deux, trois, et la signature. Ordonnée : machine bayésienne utilisée. Les différents points représentent les différentes sorties de la machine dans le même ordre que l'abscisse. Couleur : proportion de cas conduisant à la sortie dans l'ordonnée pour les gestes présentés correspondant à l'abscisse, avec la machine bayésienne fournissant une sortie certaine. Les résultats sont présentés pour un seuil de certitude T de 10%. 52

2.10 **Impact de la taille de l'ensemble de données d'entraînement sur la précision de la machine bayésienne. a** Illustration du défi des petites données. Lors de l'utilisation d'un grand ensemble de données d'entraînement représentatif, les modèles de vraisemblance peuvent être ajustés directement aux données d'entraînement. En revanche, les ensembles de données réduits peuvent ne pas être représentatifs, et l'ajustement direct d'un modèle de vraisemblance aux données peut conduire à des résultats dénués de sens. L'incorporation d'un a priori lors de l'ajustement de la vraisemblance peut cependant résoudre ce problème. Cette possibilité est un avantage important des approches bayésiennes. **b** Précision moyenne simulée des machines bayésiennes en fonction du nombre d'exemples utilisés dans l'ensemble de données d'entraînement. La précision est tracée dans une situation où les modèles de vraisemblance sont des ajustements gaussiens des données d'entraînement, et lorsqu'un a priori normal inverse chi-carré a été utilisé. Tous les résultats sont présentés en utilisant la méthode conventionnelle de calcul stochastique avec 255 cycles. Tous les résultats sont également validés par croisement, avec 20 séparations différentes entre les ensembles d'entraînement et de test. 53

3.1 **Machine bayésienne à base de memristors fabriquée. a** Photographie au microscope optique de la puce du système bayésien. **b** Détail du bloc de vraisemblance, qui comprend un circuit numérique et un bloc de mémoire avec son circuit périphérique. **c** Photographie du réseau de memristors 2T2R. **d** Image de microscopie électronique à balayage d'un memristor dans le back-end de notre processus hybride memristor/CMOS. **e** Schéma du bloc de vraisemblance présenté dans **b**. **f** *Schéma du circuit comparateur Gupta*[7], utilisé pour générer le flux de bits aléatoires de vraisemblance en comparant la probabilité lue en mémoire avec le nombre aléatoire généré par le LFSR. Pour des raisons de compacité, le circuit est présenté ici dans une version à quatre bits. Une version à *huit bits* est mise en œuvre sur la puce. **g** Schéma de l'amplificateur de détection différentiel utilisé pour lire les états binaires des memristors. **h** Schéma du convertisseur de tension, utilisé pour adapter la tension d'entrée nominale aux tensions de formation et de programmation des memristors. **i** Principe de la programmation complémentaire des memristors de la cellule de bit 2T2R. Toutes les sous-figures utilisent des codes de couleur cohérents. 64

3.2 **Réduction des effets de la variabilité des dispositifs avec l'approche 2T2R**. **a** Dans l'approche conventionnelle, une erreur se produit si un dispositif programmé en LRS a une résistance supérieure au seuil entre LRS et HRS, ou si un dispositif programmé en HRS a une résistance inférieure à ce seuil. **b** Dans l'approche 2T2R de la machine bayésienne, une erreur se produit si un dispositif programmé en LRS a une résistance supérieure à celle de son dispositif complémentaire programmé en HRS : les deux dispositifs doivent se trouver en fin de distribution pour qu'une erreur se produise. **c** Taux d'erreur de l'approche 2T2R en fonction du taux d'erreur de l'approche 1T1R, dans des simulations supposant un amplificateur de détection parfait (ligne noire) ou mesuré expérimentalement sur le circuit intégré de [8] (points bleu clair). Ligne bleue : taux d'erreur d'un code ECC SECDED utilisant le même nombre de dispositifs que notre approche 2T2R. . . . 65

3.3 **Circuit de programmation pour les matrices de mémoire de vraisemblance.** **a** Schémas détaillés de la matrice de mémoire de vraisemblance, avec son circuit de périphérie de programmation et de lecture, affichant les tensions nécessaires pour effectuer une opération SET sur le premier memristor R de la première rangée, dernière colonne. **b** Schémas des connexions de la cellule de bit 2T2R au circuit de lecture et de programmation. Deux convertisseurs de tension (convertisseur de tension conventionnel LS et convertisseur de tension à trois états TLS) et un amplificateur de détection (PCSA) sont implémentés dans chaque colonne. Un convertisseur de tension est implémenté dans chaque rangée. Le signal numérique BLEN permet de choisir entre le mode de lecture ou de programmation. **c** Tensions qui doivent être appliquées sur la ligne de bit BL, la ligne de bit inverse BLb et la ligne source SL pour le forming, la programmation d'un 'zéro' et la programmation d'un 'un' dans une cellule de bit 2T2R. **d** Niveaux de tension et chronogrammes utilisés pour les opérations de forming, de RESET et de SET. **e** Schémas au niveau du transistor du convertisseur de tension (LS) et **f** des circuits du convertisseur de tension à trois états (TLS). **g** Tableau résumant la configuration des signaux de programmation (convertisseurs de tension) pour les différentes opérations de programmation prises en charge par la matrice de mémoire (forming, programmation d'un 'zéro' et programmation d'un 'un'). 66

3.4 **Circuit de lecture pour les matrices de mémoire de vraisemblance.** Cette figure présente les schémas de circuit (**a**, **d** et **f**) et les simulations (**c**, **e**, **g** et **h**) qui expliquent l'opération de lecture de la matrice mémoire. **b** Le schéma de lecture implique une phase de précharge et une phase de décharge. **a** Schéma et **c** simulation du circuit de la phase de précharge. BL et BLb sont chargées à VDD. **d** Schéma et **e** simulation du circuit de la phase de décharge, BL et BLb sont déchargées à la masse avec une vitesse différente. **f** Schémas et **g** simulation des sorties de l'amplificateur de détection convergeant vers un état stable, pendant que BL et BLb sont complètement déchargées à la masse. 67

3.5 **Opération détaillée de la machine bayésienne.** **a** Schéma illustrant l'architecture détaillée d'un bloc élémentaire de vraisemblance. **b** Diagramme de flux des différentes opérations pour effectuer un calcul d'inférence bayésienne dans la machine bayésienne. **c** Diagramme temporel illustrant le fonctionnement de la machine bayésienne. 68

3.6 **Les différentes étapes d'un projet avec la machine bayésienne, de l'entraînement à l'inférence sur puce. a** Diagramme résumant les principales étapes d'un projet impliquant la machine bayésienne, de la construction du modèle bayésien à l'utilisation de la machine bayésienne pour effectuer une inférence sur puce. **b** Illustration des différentes étapes de normalisation, de discrétisation et de quantification des facteurs de vraisemblance, nécessaires avant la programmation des tableaux de mémoire de vraisemblance sur puce. 69

3.7 **Mesures des vraisemblances stockées dans le démonstrateur. a** Motifs mesurés avant l'étape de formation (forming). Les résultats des mesures semblent aléatoires. **b** Motifs destinés à être stockés. **c** Motifs mesurés immédiatement après la programmation. **d** Motifs mesurés après cinq mois, pendant lesquels le démonstrateur a été conservé à température ambiante. Les images **a**, **c**, **d** sont présentées en couleurs pour exprimer qu'elles représentent des mesures effectuées sur puce, tandis que l'image **b** est présentée en noir et blanc pour exprimer qu'elle représente un motif prévu. 71

3.8 **Mesures de la machine bayésienne à base de memristors fabriquée. a** Sortie de la machine bayésienne : probabilité postérieure mesurée en fonction de la valeur attendue de la loi de Bayes. Les différents points correspondent à des entrées d'observations aléatoires. Les différentes lignes sont regroupées dans le même graphique. Les points sont obtenus avec différentes tensions d'alimentation VDD comprises entre 0,5 et 1,2 volts. Ce graphique est obtenu avec des *seeds* LFSR non optimaux. Les probabilités mesurées sont obtenues en moyennant les mesures expérimentales sur toute la période LFSR (255 cycles). **b** Identique à **a**, en utilisant des *seeds* LFSR optimaux. Les symboles indiquent quelle ligne de la machine bayésienne a été utilisée (cercle, triangle vers le haut, triangle vers le bas, carré : première, deuxième, troisième et quatrième ligne). 73

3.9 **Systèmes de tests des deux puces bayésiennes. a** Installation sous la station sous pointes au laboratoire IM2NP à Marseille et **b** Installation dans un package JLCC52 et PCB, réalisée au C2N. 74

3.10 **Mesure expérimentale de la perturbation due à la lecture** sur une matrice de mémoire de vraisemblance, avec une valeur VDD de 1,2 volts. Même après 5,7 millions d'opérations de lecture de l'ensemble de la matrice, aucune erreur n'est observée. 75

3.11 **Étapes de la simulation de consommation énergétique des puces constituées de memristors.** 77

3.12 **Application de la machine bayésienne à une tâche pratique de reconnaissance de gestes. a** Configuration avec une unité de mesure inertielle utilisée pour enregistrer l'ensemble de données de reconnaissance de gestes. **b** Masques de la conception de la machine bayésienne placée et routée utilisée pour effectuer l'analyse de la reconnaissance de gestes au niveau de la conception. **c** Consommation d'énergie du système (consommation dynamique et matrices de mémoire) au cours des trois phases de calcul : chargement des *seeds* dans le LFSR, lecture des mémoires et inférence réelle de 255 cycles. **d** Consommation d'énergie des points importants du système pendant la phase d'inférence pour 255 cycles. **e** *Précision moyenne* en fonction du nombre de cycles pour deux types de calcul : en utilisant une méthode économe en énergie en ne prenant en compte que le premier '1' pour la décision (en rouge) et en utilisant le calcul stochastique conventionnel en utilisant le nombre maximum de '1' pour la décision (en bleu). Les ombres autour du graphique montrent un écart-type de la précision moyenne sur les dix sujets. **f** Consommation d'énergie pendant la phase d'inférence en fonction de la précision pour la reconnaissance de gestes pour les deux méthodes. Les étoiles correspondent au même point dans les graphiques **e** et **f**. Toutes les valeurs d'énergie sont données pour une tension d'alimentation de 1,2 volts. . . 82

4 Machine Bayésienne III, les deux modes! 83

4.1 **Fonction logarithmique transformant la probabilité en entier 8 bits**, pour trois valeurs de m : 8, 32, 64. 86

4.2 **Architecture générale de la machine bayésienne logarithmique. a** Schéma simplifié de la machine bayésienne logarithmique mise en œuvre. **b** Schéma simplifié de la likelihood comprenant les additionneurs. 87

4.3 **Mise en œuvre expérimentale de la classification des phases du sommeil tout au long de la nuit. a** Entrées et sorties de la machine bayésienne logarithmique utilisée pour la classification des phases du sommeil. **b** Modèle de réseau bayésien implémenté sur la machine bayésienne logarithmique utilisée pour la classification des phases du sommeil. 89

4.4 **Évaluation de la précision et de la consommation d'énergie des machines bayésiennes stochastiques et logarithmiques. a,b** Précision de la machine stochastique pour **a** la reconnaissance des gestes et **b** la classification des phases du sommeil, en fonction du nombre de cycles d'horloge, en utilisant le calcul stochastique conventionnel ou l'approche "power-conscious". La précision de la machine logarithmique (un cycle d'horloge) est tracée comme référence. **c,d** Consommation d'énergie en fonction de la précision pour **c** la reconnaissance des gestes et **d** la classification des phases du sommeil, en utilisant toutes les approches considérées dans a et b. Cette figure est obtenue en associant plusieurs méthodologies de simulation. Les barres d'erreur/ombres représentent un écart type. 90

4.5 **Évaluation de la Résilience au taux d'erreur de bit de memristor de la machine bayésienne stochastique et logarithmique.** Précision des machines bayésiennes stochastiques (utilisant le calcul conventionnel et "power-conscious") et logarithmiques en fonction des taux d'erreur de bit de memristor, pour **a** la reconnaissance des gestes et **b** les tâches de classification des phases du sommeil. Cette figure est obtenue en utilisant la simulation de Monte Carlo. Les barres d'erreur/ombres représentent un écart type. 93

4.6 **Puce bayésienne logarithmique et son système de test. a** Image de microscopie optique de la puce de la machine fabriquée. **b** Photographie de la configuration du test de la machine bayésienne logarithmique. 94

4.7 **Résulat de mesures de la puce bayésienne** Précision mesurée expérimentalement de la machine bayésienne logarithmique, moyennée sur 1000 segments de cinq secondes. La mesure a été répétée pour différentes tensions d'alimentation VDD. 95

4.8 **Intégration des machines bayésiennes avec un cœur RISC-V. a** Configuration du test expérimental avec machine bayésienne et cœur RISC-V. **b** Schéma de l'intégration de la machine bayésienne dans un coeur RISC-V. **c** Précision mesurée de la sortie de la machine bayésienne logarithmique (moyenne sur 4 096 entrées), en fonction de la tension d'alimentation, pour différents courants de compliance SET (contrôlés par la tension de grille du transistor de sélection). 97

Introduction

On m'attribue le mérite d'être l'un des plus grands travailleurs et peut-être le suis-je, si la pensée est considérée comme un équivalent du travail, car j'y ai consacré presque toutes mes heures d'éveil. Mais si l'on interprète le travail comme une activité définie dans un temps déterminé et selon une norme rigide, alors je suis peut-être le pire des paresseux. Tout effort imposé demande un sacrifice de l'énergie vitale. Je n'ai jamais payé un tel prix. Au contraire, je me suis épanoui dans mes pensées.

Nikola Tesla

“ *Enfoncez-vous bien dans votre fauteuil, prenez un paquet de pop-corn et laissez-vous maintenant embarquer dans ce voyage merveilleux.* ”

Dans le panorama technologique actuel, l'intelligence artificielle (IA) s'est affirmée comme une force centrale dans une multitude de domaines, catalysant des avancées significatives dans des secteurs aussi variés que l'analyse de données, la médecine, et la robotique. Les systèmes IA contemporains, y compris les modèles génératifs avancés tels que ChatGPT, Llama, Claude dans la génération de texte ou encore Midjourney, Dall-E dans la génération d'images, opèrent principalement dans des environnements cloud, centralisant ainsi une quantité colossale de données. Cette centralisation dans le cloud, bien qu'elle facilite l'accès et la manipulation des données, engendre des préoccupations croissantes en matière de sécurité des données. Comme le souligne Castets-Renard [9], cette centralisation peut créer des vulnérabilités significatives, exposant les utilisateurs à des risques de violations de données et d'attaques cybernétiques.

Dans ce contexte, l'adoption de l'IA dans les systèmes embarqués tels les smartphones, smartwatch et autres appareils mobiles se présente comme une alternative prometteuse. Cela permettrait une décentralisation des données et une réduction des risques associés à la centralisation. Cependant, cette transition est loin d'être sans obstacles. Un des défis majeurs réside dans la consommation massive d'énergie associée aux opérations de l'IA, un problème qui est exacerbé dans les environnements mobiles où les ressources sont limitées. En effet, les IA ont besoin d'être entraînées et cet entraînement consomme énormément de ressources et d'énergie. Un exemple dans le cadre de l'entraînement, récemment Meta a publié les données pour son modèle de langage Llama-2. Pour l'entraînement de leur plus petit modèle avec sept milliards de paramètres (Llama-2 7B), il a fallu 184320 heures sur des cartes graphiques A100-80GB qui ont une puissance maximale de 400 W. Ce qui donne une consommation énergétique de 73.7 MWh soit l'équivalent de la consommation annuelle moyenne de 33 foyers français[10]. Au vu de ces chiffres, l'entraînement n'est pas réalisable sur un appareil mobile avec une capacité limitée. Cependant il est possible de réaliser l'entraînement en dehors de l'appareil et de transférer le modèle obtenu puis de réaliser uniquement l'inférence sur l'appareil. Un autre exemple que j'aimerais vous proposer et l'utilisation de l'inférence pour générer des images. La figure 1 a été générée en utilisant le modèle SDXL 1.0 [1] sur mon ordinateur avec une carte graphique RTX Titan qui consomme au maximum 280 W. Elle a été générée en 25.2 s ce qui donne une consommation de 1.96 Wh ce qui reste énorme pour faire juste une inférence. De plus, pour générer cette image, il faut une puissance de calcul appropriée. Cela signifie que le matériel utilisé doit être suffisamment performant et adapté pour traiter les données et exécuter les algorithmes requis par l'inférence, faute de quoi, le processus peut être gravement compromis ou même impossible à réaliser. Ce niveau de performance n'est généralement pas disponible dans des appareils aux capacités limitées, tels que les appareils mobiles.

En plus du problème énergétique, l'IA actuelle est souvent critiquée pour son manque de transparence et d'explicabilité, ce qui peut entraver son adoption dans des domaines où la compréhension des processus décisionnels est cruciale comme le domaine médical ou encore la conduite autonome. Dans cette optique, une méthode plus explicable comme l'inférence

Figure 1: **Image de moi présentant ma thèse, générée par Stable Diffusion.** Entraînement du modèle en utilisant des photos de moi et l'environnement kohya_ss (https://github.com/bmaltais/kohya_ss) puis générée en utilsant l'environnement AUTOMATIC1111 (https://github.com/AUTOMATIC1111/stable-diffusion-webui) avec le modèle SD-XL [1] en 60 itérations.

bayésienne permettrait de devenir une solution prometteuse. En adoptant des principes probabilistes dans l'IA, il est possible de développer des modèles qui non seulement sont capables de rationaliser leurs décisions, mais aussi d'offrir une plus grande transparence dans leurs opérations.

C'est dans cette optique de consommation énergétique que je vais commencer cette thèse avec dans le premier chapitre une exploration de l'architecture des circuits actuels puis de l'avancée des nouvelles architectures inspirées sur le fonctionnement du cerveau qui m'amènera à discuter des mémoires émergentes, en mettant un accent particulier sur les memristors et leur potentiel pour révolutionner le calcul en mémoire.

Dans le deuxième chapitre, je présenterai la machine bayésienne. Depuis l'explication du fonctionnement jusqu'à une application réelle de reconnaissance de gestes en passant par l'architecture de cette machine bayésienne. Je décomposerai les principes fondamentaux qui sous-tendent cette approche, en examinant de près les techniques d'approximation et de multiplication qui sont centrales à son fonctionnement. Cette partie se penchera également sur les applications pratiques de cette approche dans la conception des circuits, explorant les possibilités offertes pour la reconnaissance des gestes et évaluant la performance de ces systèmes dans des contextes à faible volume de données et surtout du caractère explicable de la machine.

Le troisième chapitre est consacré sur l'aspect conception et réalisation de la machine bayésienne où je détaillerai les étapes clés du processus de fabrication, en mettant en lumière les innovations récentes dans la structuration et la programmation de cette machine. J'explorerai

également les défis et les opportunités associés à la mesure et à la caractérisation de ces systèmes, avant de conclure avec une discussion sur les perspectives d'amélioration de l'efficacité énergétique.

Enfin, le quatrième chapitre est consacré à une comparaison entre deux architectures de calcul pour une machine bayésienne, je ferai la comparaison tant en terme d'application, sur la reconnaissance de gestes et également sur une application plus médicale qui est la reconnaissance des cycles du sommeil, qu'en terme énergétique. Puis, je présenterai une intégration de cette puce dans un système composé d'un coeur RISC-V. Pour terminer, je discuterai brièvement d'une nouvelle puce regroupant les deux méthodes de calcul.

J'ai également eu l'opportunité de participer à d'autres projets décrits dans les annexes de cette thèse. Le premier (annexe A) consiste à la conception d'une puce bimodale (analogique et numérique) constituée d'un tableau de 64x128 memristors inclus dans une structure 2T2R [11]. Cette puce permet d'effectuer des opérations de lecture et d'écriture de manière numérique mais également une opération de lecture analogique permettant de mesurer l'état de résistance des memristors. Cette plateforme permet d'optimiser les conditions de lecture et d'écriture des memristors, ainsi que de développer et de tester des concepts neuromorphiques innovants basés sur les memristors. Dans ce projet, j'ai conçu la puce la partie numérique de la puce et effectué les tests de vérifications de fonctionnement avant la fabrication.

Dans le deuxième projet (annexe B), nous avons développé un réseau neuronal binarisé robuste, composé de 32 768 memristors et alimenté par une cellule solaire optimisée, destiné à des applications sur des systèmes embarqués [12]. La conception unique du circuit permet un calcul numérique efficace et résilient proche de la mémoire, éliminant le besoin de compensation ou de calibration et assurant une performance optimale sous diverses conditions. Il a démontré une performance d'inférence comparable à une alimentation de laboratoire sous une forte illumination et a maintenu sa fonctionnalité avec une précision légèrement réduite dans des conditions de faible éclairage. Ma participation dans ce projet consiste à l'évaluation de la consommation énergétique de la puce en simulation.

Enfin, le troisième projet (annexe C) présente une méthode d'apprentissage automatique qui exploite la variabilité des memristors pour mettre en œuvre l'échantillonnage de Markov chain Monte Carlo dans un réseau fabriqué de 16 384 dispositifs, configuré comme un modèle d'apprentissage machine bayésien [13]. Nous expérimentons cette approche pour effectuer la reconnaissance de tissus malins et la détection d'arythmie cardiaque, et, à l'aide d'un simulateur calibré, nous abordons la tâche d'apprentissage par renforcement de cartpole. Notre méthode démontre une robustesse face à la dégradation des dispositifs à dix millions de cycles d'endurance et, selon des simulations au niveau du circuit et du système, l'énergie totale requise pour entraîner les modèles est estimée à l'ordre des microjoules, ce qui est notablement inférieur aux approches basées sur les technologies CMOS. J'ai effectué les estimations énergétiques de la puce en simulation sur l'application du cartpole.

J'ai effectué cette thèse en collaboration avec différentes équipes, notamment l'équipe d'Elisa

Vianello du CEA-LETI à Grenoble pour la fabrication des memristors et l'équipe de Jean-Michel Portal et Marc Bocquet de l'IM2NP pour la conception des puces. Au cours de ma thèse, j'ai également collaboré avec les étudiants suivants du C2N : Kamel-Eddine Harabi, Tifenn Hirtzlin, Marie Drouhin, Atreya Majumdar, Adrien Renaudineau, Théo Ballet, Adrien Pontlevy et Fadi Jebali tant dans la conception des puces que dans leurs tests et leurs applications.
La puce Bayésienne stochastique (Chapitres 2 et 3) a été conçue avant ma thèse par Kamel-Eddine Harabi et Tifenn Hirtzlin. J'ai contribué à son test électrique, développé la version passée à l'échelle du système et effectué toutes les analyses sur cette version passée à l'échelle du système, notamment toutes les études concernant la reconnaissance de gestes (Chapitres 2 et 3). J'ai également développé la méthodologie pour l'évaluation énergétique de la puce. J'ai développé le concept de la machine Bayésienne logarithmique. J'ai conçu sa démonstration ASIC (en collaboration avec Kamel-Eddine Harabi), le PCB pour son test et réalisé sa caractérisation électrique (Chapitre 4). Le travail sur les cycles du sommeil a été réalisé par Adrien Pontlevy, sous ma supervision conjointe avec celle de mon directeur de thèse (Chapitre 4). L'intégration dans le RISC-V a été effectuée par le stagiaire Théo Ballet (Chapitre 4). J'ai conçu la grande machine Bayésienne multimode (en collaboration avec Kamel-Eddine Harabi) et développé son PCB de test (Chapitre 4). J'ai conçu la plateforme de tests multimodes analogique/numérique en collaboration avec Kamel-Eddine Harabi et développé son PCB de tests (Annexe A). J'ai réalisé l'analyse énergétique d'une puce de l'IM2NP implémentant un réseau de neurones (Annexe B) et d'un système d'échantillonnage MCMC utilisant la physique des memristors (Annexe C).

Chapter 1

Énergie et Mémoires résistives : Un voyage inattendu

L'association des mémoires résistives et du calcul proche-en mémoire pour diminuer la consommation des circuits

> "L'Homme ne trouvera jamais une invention plus belle, plus simple ou plus directe que la nature, car dans ses inventions rien ne manque et rien n'est excessif."
>
> Léonard de Vinci

1.1 Vers une optimisation de l'efficacité énergétique des circuits...

L'efficacité énergétique des circuits intégrés est une préoccupation majeure dans le domaine de l'électronique, de l'informatique et maintenant de l'intelligence artificielle. La miniaturisation des transistors et l'augmentation de leur densité sur les puces ont permis d'augmenter considérablement les performances des circuits, mais ont également posé de nouveaux défis en termes de gestion de l'énergie et de la chaleur.

L'architecture de Von Neumann, qui est à la base de la plupart des ordinateurs modernes, a joué un rôle clé dans cette évolution. Cette architecture, qui sépare le stockage des données et leur traitement, a permis de concevoir des machines universelles capables d'exécuter n'importe quel programme stocké sous forme de données. Cependant, le transport des données entre le processeur et la mémoire consomme une grande partie de l'énergie utilisée par les ordinateurs modernes, ce qui limite leur efficacité énergétique [14].

Face à ces défis, de nouvelles architectures sont explorées, inspirées par le fonctionnement du cerveau humain. Ces architectures, dites neuromorphiques, cherchent à imiter la manière dont les neurones et les synapses du cerveau traitent et stockent l'information de manière distribuée et parallèle. Elles promettent une efficacité énergétique bien supérieure à celle des architectures traditionnelles, en particulier pour les tâches liées à l'intelligence artificielle et à l'apprentissage machine [15].

1.1.1 Évolution des circuits et de l'architecture de Von-Neumann

1.1.1.1 Évolution des circuits

J'aimerai faire l'analogie avec les débuts de l'aviation, seulement 65 ans séparent le premier vol en avion des frères Wright en 1903 jusqu'au premier pied posé par Neil Armstrong sur la Lune en 1969 [16], de la même manière, en seulement 75 ans, nous sommes passés de l'invention du transistor en 1947 [17, 18] qui a remplacé les vieux tubes à vide encombrants et énergivores. Puis, en passant par l'utilisation des nouveaux matériaux semiconducteurs comme le silicium [19] à des systèmes électroniques complexes capables de réaliser des milliards d'opérations par seconde[20]. Ces avancées ont permis l'émergence de technologies que nous utilisons quotidiennement, comme les ordinateurs, les smartphones et l'Internet. Tout comme l'aviation a transformé notre façon de voyager et de comprendre le monde, l'évolution des transistors a révolutionné notre façon de communiquer, de travailler et de vivre

La croissance exponentielle dans la miniaturisation des transistors a été prédite par Gordon Moore en 1965 [21], la loi de Moore formule une multiplication par deux des transistors dans une puce tous les un an et demi permettant ainsi l'augmentation des performances et l'efficacité de la consommation énergétique de celle-ci. Cette loi a été respectée pendant plus

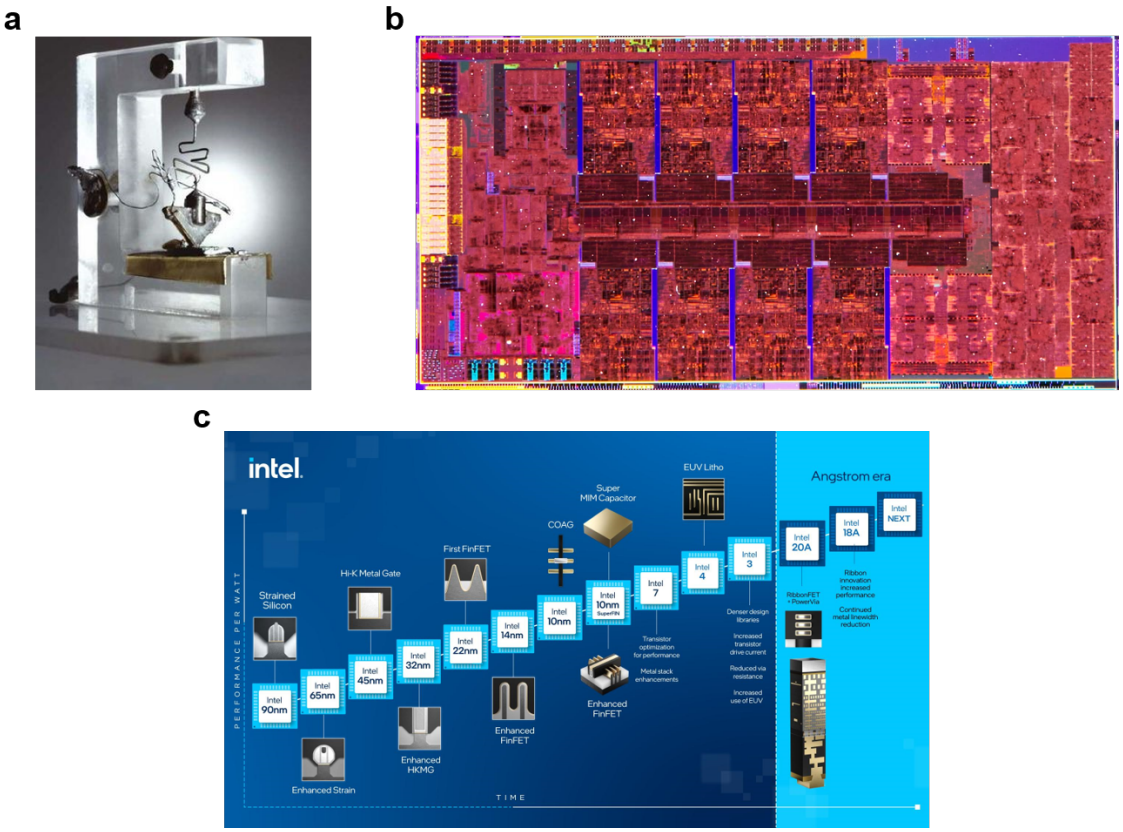


Figure 1.1: **Évolution du transistor : du premier aux 1000 milliards sur puce.** **a** Le premier transistor, développé par Walter Brattain et John Bardeen en 1947, source:Alcatel-Lucent USA, Inc **b** Photo du die du i9-12900K de Intel, source:Intel et **c** Évolution de la technologie de fabrication d'Intel, du passé au futur, source:https://www.intel.com/content/www/us/en/newsroom/resources/moores-law.html

de 50 ans [22]. Elle a été le moteur de la miniaturisation des transistor avec la recherche de nouveaux designs comme le MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor) [23, 24] qui devient le design le plus utilisé dans les puces modernes car il permet une commutation plus rapide et surtout une consommation réduite par rapport à ses prédécesseurs. Il a subi de nombreuses transformations depuis sa version planaire, le Planar FET, qui a atteint une taille minimale de 28 nm [25, 26]. Cependant, la diminution de la taille des transistors a entraîné des difficultés croissantes dans la gestion du courant, provoquant l'émergence de courants de fuite et une hausse de la consommation d'énergie. En réponse à ces défis, des architectures de transistors innovantes ont été conçues, comme le FD-SOI (Fully Depleted Silicon On Insulator) [27, 28], ou le FinFET [29, 30] qui permettent de passer sous la barre des 28 nm. Malgré ces avancées technologiques, la limite diminue toujours mais sans pouvoir passer la barre des 3 nm avec ces architectures [31]. Pour vraiment pouvoir passer cette barre, les grands fabricants recherchent d'autres architectures plus complexes comme celle des transistors à grille

enrobante (Gate-All-Around FET ou GAA-FET) [32], technologie qui a déjà été utilisée par Samsung pour son architecture 3 nm [33]. Cette technologie, où le matériau du canal est entièrement enveloppé par le matériau de la grille, permet un contrôle plus précis du courant dans le canal. Cela se traduit par une amélioration de l'efficacité énergétique et des performances du transistor. Cependant, la complexité accrue de la fabrication des transistors GAA peut entraîner une augmentation des coûts de production. De plus, la conception de circuits utilisant des transistors GAA peut présenter des défis en raison de la complexité inhérente à l'architecture du transistor. L'ensemble de toutes ces complications dues à l'architecture a fait que seuls quelques grands fondeurs continuent la recherche et le développement passé cette limite des 3 nm [34–36]. Plus récemment, comparable aux transistors GAA, l'architecture CFET (Complementary FET) est apprarue, elle combine un transistor nMOS et un transistor pMOS dans une configuration verticale, ce qui permet une commutation plus rapide et une réduction de la consommation d'énergie [37, 38]. En outre, l'architecture CFET permet une densité de transistors plus élevée, ce qui est essentiel pour maintenir la loi de Moore; mais comme toutes les nouvelles techniques avancées, la fabrication est complexe et coûteuse.

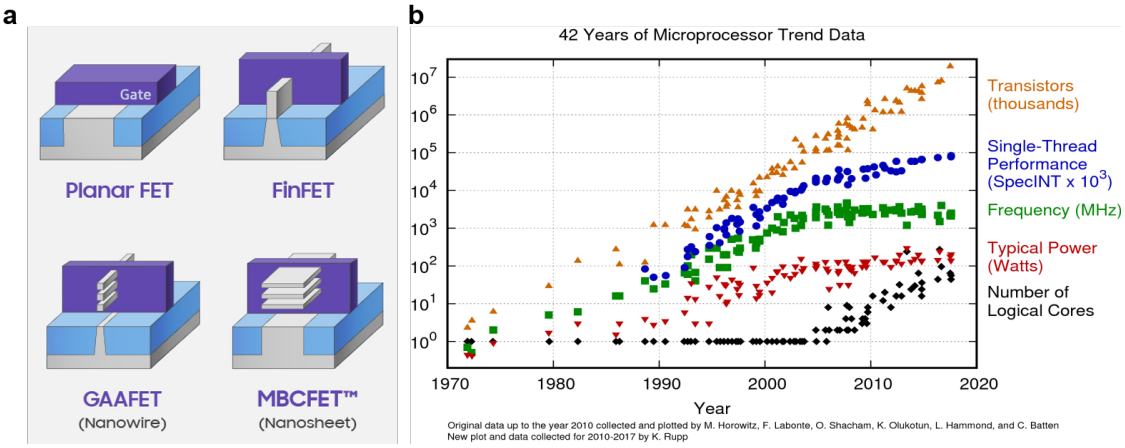


Figure 1.2: **Évolution du MOSFET. a** Évolution de l'architecture du MOSFET, source : Samsung, https://www.samsungfoundry.com/foundry/homepage.do **b** Graphe illustrant la fin de l'augmentation exponentielle des performances; la loi de Moore va sûrement se terminer aussi, tiré de [2]

Cependant, diminuer la taille des transistors n'est pas une fin en soi. Pour réaliser une puce, il faut que l'on puisse réaliser des circuits complexes. C'est là qu'interviennent les langages de descriptions et les différents outils associés à la création de circuits[39]. Le problème de comment réaliser un circuit s'est posé dès le milieu des années 80 : les langages VHDL [40] (Very high speed integrated circuit Hardware Description Language) et Verilog ont été créés et sont devenus la norme pour la description des circuits. Plus récemment, le SystemVerilog est apparu pour améliorer la modélisation de systèmes et la vérification de conceptions. Il introduit des types de données plus complexes, une meilleure gestion des interfaces, et des fonctionnalités de programmation orientée objet. Décrire le fonctionnement du système, c'est bien, mais

il faut ensuite pouvoir l'implémenter sur du silicium, c'est à ce moment que sont utilisés les outils de synthèse, placement, routage des circuits qui vont transformer le code HDL en portes logiques pour l'étape de synthèse puis au placement de ces portes et leurs connexions sur une puce qui pourra être fabriquée. Les systèmes devenant de plus en plus complexes, il a fallu passer d'une conception à la main à une conception assistée par ordinateur [41]. De nos jours, pour concevoir des circuits dédiés à l'intelligence artificielle toujours plus performants et complexes, les concepteurs ont besoins d'outils de plus en plus avancés et complexes. L'innovation ne se limite pas à la miniaturisation des transistors, mais s'étend également à la manière dont ces transistors sont utilisés pour construire des architectures de circuits plus efficaces. Dans ce contexte, l'intelligence artificielle elle-même est devenue un outil précieux pour la conception de circuits. L'IA peut être utilisée pour automatiser et optimiser la conception de circuits, permettant ainsi à l'IA de "construire l'IA"[42]. Cette approche pourrait ouvrir la voie à des avancées significatives dans la performance et l'efficacité des circuits dédiés à l'IA et "restaurer la magie du design", comme l'a récemment exprimé Jan Rabaey, l'auteur de l'ouvrage de référence "Digital Integrated Circuits : A Design Perspective" qui a formé plusieurs génération de concepteurs [42].

1.1.1.2 Architecture de von Neumann

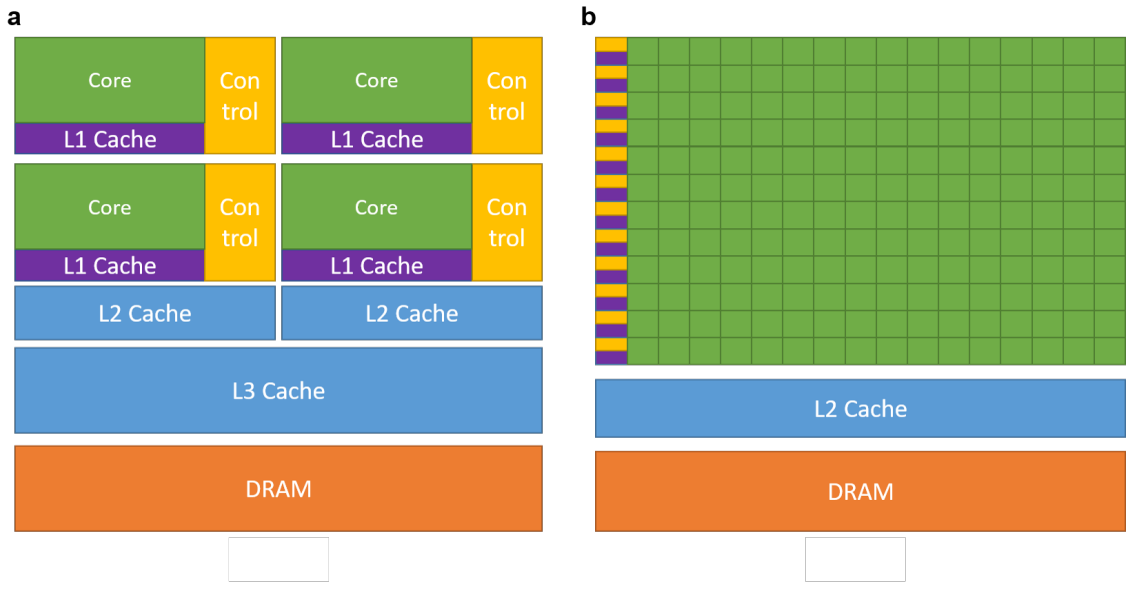


Figure 1.3: **Architecture des ordinateurs actuels. a** Schéma d'une architecture d'un CPU simplifiée et **b** d'un GPU, adapté de la documentation CUDA de NVIDIA

L'architecture des ordinateurs a connu une transformation significative depuis l'ENIAC, le premier ordinateur numérique électronique à grande échelle. Malgré ses défis, l'ENIAC a posé les bases de l'informatique moderne [43]. L'architecture de von Neumann, qui est devenue la norme pour la plupart des ordinateurs à usage général, a été un jalon majeur dans cette

évolution. Elle comprend un processeur central (CPU), une mémoire, et des dispositifs d'entrée et de sortie, tous reliés entre eux [44].

Avec le temps, les systèmes informatiques sont devenus plus compacts et plus puissants. Le système de codage rapide IBM 701 a été l'un des premiers à utiliser un calculateur à virgule flottante à trois adresses. Cette conception permettait à la machine d'effectuer des opérations arithmétiques sur des nombres à virgule flottante en utilisant trois adresses mémoire : une pour chaque opérande et une pour le résultat. Cela a grandement amélioré l'efficacité du traitement de l'information, car il était possible d'effectuer des opérations complexes en moins d'étapes qu'avec des systèmes antérieurs [45]. Plus tard, l'architecture du système IBM System/360 a introduit des concepts tels que le multiprogramming et le time-sharing, optimisant l'utilisation des ressources informatiques [46].

Le développement du microprocesseur chez Intel a marqué une étape importante, réduisant la taille des ordinateurs tout en augmentant leur puissance de traitement, ouvrant ainsi la voie à l'ère de l'informatique personnelle [47]. L'approche quantitative de l'architecture des ordinateurs introduite par David Patterson et John L.Hennessy a souligné l'importance du parallélisme et de la hiérarchie de la mémoire pour améliorer les performances des systèmes informatiques [48].

Parallèlement à ces développements, les unités de traitement graphique (GPU) ont émergé, offrant une puissance de calcul parallèle qui dépasse largement celle des CPU. Les GPU, en plus d'être des moteurs graphiques puissants, sont des processeurs hautement parallèles programmables, offrant une bande passante arithmétique et mémoire particulièrement élevée [49, 50].

L'ère moderne de l'architecture des ordinateurs, souvent décrite comme un nouvel âge d'or, est alimentée par la révolution de l'apprentissage automatique. Cette révolution, qui a transformé le traitement de la vision, la parole, la compréhension du langage, et de nombreux autres domaines, a également conduit à l'émergence des unités de traitement tensoriel (TPU). Développées par Google, ces unités ont été conçues pour accélérer l'apprentissage automatique, en particulier l'apprentissage profond. Elles offrent des performances élevées et une efficacité énergétique pour les charges de travail de l'apprentissage profond, marquant ainsi une étape importante dans l'évolution de l'architecture des ordinateurs par l'émergence d'unités de calcul plus spécialisées [51, 52].

L'architecture de Von Neumann, bien que fondamentale dans la conception des ordinateurs modernes, présente des défis majeurs en termes de consommation énergétique. Le "goulot d'étranglement de Von Neumann" est un problème inhérent à cette architecture, où la séparation entre le stockage et le traitement des données conduit à une consommation d'énergie excessive lors du transfert de données entre la mémoire et le processeur [53]. Ce goulot d'étranglement est dû à la limitation de la bande passante entre le processeur et la mémoire, ce qui oblige le processeur à attendre pendant que les données sont transférées. Cela entraîne une inefficacité dans l'utilisation de la puissance de calcul du processeur, ce qui se traduit par une consom-

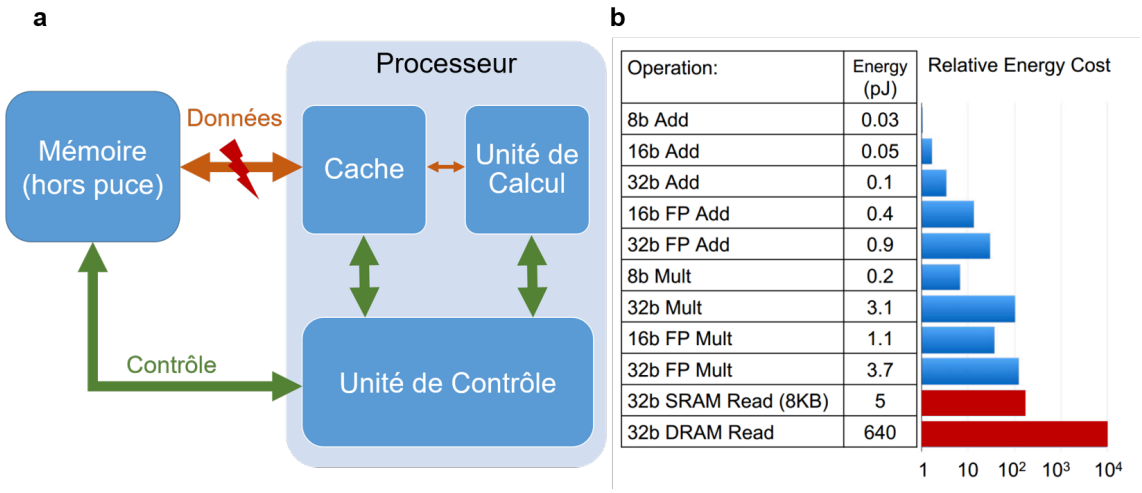


Figure 1.4: **Limites de l'architecture de Von Neumann. a** Schéma d'une architecture d'un CPU simplifiée avec le goulot d'étranglement de Von Neumann à cause de la séparation physique de la mémoire et du calcul et **b** Coûts énergétiques pour différentes opérations arithmétique comparés aux coûts pour l'accès mémoire, tiré de [3]. L'accès à la DRAM est 4 ordres de grandeur plus énergivore que l'addition sur 8 bits

mation d'énergie inutile comme le montre la Fig. 1.4b. De plus, cette inefficacité énergétique est exacerbée par l'augmentation de la demande de calculs de haute performance dans des domaines tels que l'intelligence artificielle et le big data. Dans ces domaines, la quantité de données à traiter est si grande que le transfert de données entre le processeur et la mémoire devient un facteur limitant majeur de la performance énergétique du système[3, 54].

En conclusion, la miniaturisation des transistors a été principalement motivée par le désir d'améliorer les performances des circuits intégrés tout en réduisant leur coût et leur consommation d'énergie. Cependant, la miniaturisation et l'architecture de Von Neumann semblent avoir atteint leurs limites [55] pour les nouvelles tâches d'intelligence artificielle. Il faut donc se tourner vers d'autres architectures pour pouvoir continuer d'augmenter exponentiellement les performances des futures puces de calcul [56].

1.1.2 Architecture inspirée par le Cerveau

Comme Clément Ader et ses avions aux allures de chauve-souris et les frères Wright et leur système de pilotage se sont inspirés du vol des oiseaux pour créer le premier avion, nom donné par Ader du latin *avis* signifiant oiseau, les chercheurs se sont inspirés de ce que la nature a fait de mieux en matière d'unité de calcul : le cerveau.

Le cerveau humain est une structure complexe et fascinante, composée d'environ 10 000 milliards de cellules neuronales interconnectées [57]. Ces neurones se distinguent en différentes catégories, chacune possédant des éléments de base similaires mais disposés de manière unique. Il est important de noter que chaque neurone possède près de 10 000 synapses, ce qui suggère

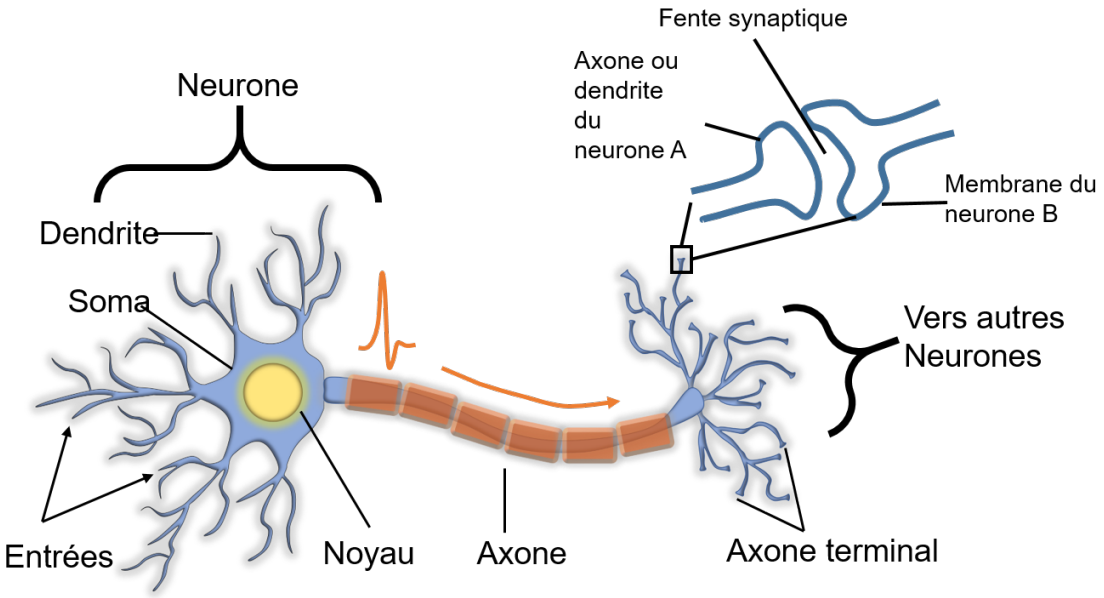


Figure 1.5: **Schéma d'un neurone.** Les signaux neuronaux entrent par les dendrites, des branches vastes et épaisses qui émergent du soma, et sont recouvertes de synapses. Ces signaux, appelés potentiels d'action, transitent ensuite vers d'autres neurones via des axones. Ces derniers, capables de s'étendre sur de grandes distances, peuvent atteindre plus d'un mètre de long et se ramifient de nombreuses fois pour permettre une communication efficace. Un seul neurone peut ainsi intégrer plus de 10 000 synapses. (Schéma adapté de Shuai Li)

que le cerveau peut aussi être envisagé comme un ensemble de synapses avec un nombre réduit de neurones imbriqués à l'intérieur, formant une multitude de connexions câblées. D'un point de vue architectural, on peut donc appeler le cerveau un système de calcul en mémoire.

Les neurones, ces cellules fondamentales du système nerveux, échangent des informations via des structures filiformes appelées axones. Ces axones permettent à un neurone de transmettre des impulsions électriques, ou "spikes", vers d'autres neurones ou vers des cellules cibles spécifiques dans diverses régions du cerveau. La connexion entre deux neurones se fait au niveau des synapses, qui sont des points de contact entre l'axone du neurone émetteur (pré-synaptique) et les dendrites du neurone récepteur (post-synaptique). Au sein du neurone récepteur, l'information provenant de diverses sources est intégrée dans le soma, la partie centrale du neurone, via les dendrites (Fig. 1.5). Plusieurs modèles ont été développés pour décrire le comportement des neurones en fonction des potentiels d'action qu'ils reçoivent. Le modèle le plus simple, appelé "integrate and fire", propose que le soma somme les informations reçues sous forme de spikes au niveau des dendrites, ce qui augmente son potentiel membranaire. Lorsque ce potentiel atteint un certain seuil, le neurone génère à son tour un potentiel d'action. Cependant, ce modèle est simpliste. Un modèle plus sophistiqué, appelé "leaky integrate and fire", est souvent utilisé pour simuler le comportement des neurones biologiques. Ce modèle ajoute un terme de fuite au modèle "integrate and fire", qui tend à ramener le potentiel

Caractéristiques	Ordinateur	Cerveau
	Différences	
Tâches	Arithmétiques	Cognitives
Vitesse	Rapide : ≈ 1 ns	Lente : 1 ms
Dispositifs	Transistors déterministes	Synapses bruitées et stochastiques
Précision	Flottant 32/64 bits (nombres réels)	Nombres réels imprécis
Température de fonctionnement	5°C – 80°C	36°C – 38°C
Types d'opérations	Séquentielles	Massivement parallèles
Communication	Synchrone, basée sur une horloge	Totalement asynchrone
Mémoire	Séparée du calcul	Calcul en mémoire (Synapse-Neurone)
Énergie	$\approx 10^3$ W	20 W
Structure	2D	3D
	Similarités	
Évolution	Recherche d'optimisation énergétique	
Communication	Signal électrique binaire	
Structure	Composée d'éléments à l'échelle nanométrique	

Table 1.1: Tableau récapitulatif des différences et simularités entre le cerveau et un ordinateur, tiré de [64]

membranaire à sa valeur de repos avec le temps. Un modèle, plus complexe, est le modèle de Hodgkin-Huxley [58], qui a valu à ses créateurs le prix Nobel en 1963. Ce modèle offre une représentation plus précise des potentiels d'action, mais sa complexité le rend difficile à utiliser pour simuler de grands réseaux de neurones. En ce qui concerne les synapses, leur petite taille rend difficile leur étude individuelle, et par conséquent, notre compréhension de leur fonctionnement est encore partielle. Nous savons cependant qu'il existe deux types de synapses : les synapses excitatrices, qui augmentent le potentiel de membrane, et les synapses inhibitrices, qui inhibent la potentialisation [59]. En outre, il est difficile de déterminer précisément la valeur des poids synaptiques, c'est-à-dire la force de la connexion entre deux neurones. Ces poids sont souvent considérés comme imprécis et sujets à du bruit [60].

En ce qui concerne la communication entre les neurones, nous savons qu'elle est asynchrone, c'est-à-dire qu'il n'y a pas de signal global entre les neurones qui contrôle l'information transmise. De plus, la communication est intrinsèquement binaire, puisque les seuls signaux transmis sont des spikes [61, 62]. Certaines théories suggèrent qu'une forme de communication non entièrement binaire pourrait être possible, impliquant des rafales de spikes [63]. Ce fonctionnement rappelle le calcul stochastique que je vais développer dans les chapitres 2 et 3. De plus, le cerveau est capable d'apprendre et de s'adapter grâce à un processus appelé plasticité synaptique. Cela signifie que les connexions entre les neurones peuvent se renforcer ou s'affaiblir en fonction de l'activité neuronale. Par exemple, si deux neurones sont souvent activés ensemble, la connexion entre eux peut se renforcer, ce qui facilite la transmission de l'information entre ces neurones à l'avenir [63]. C'est ce processus qui est à la base de notre capacité à apprendre et à mémoriser de nouvelles informations.

En s'inspirant du fonctionnement du cerveau, les chercheurs ont exploré des approches différentes, tant au niveau des architectures (hardware) que des algorithmes (software). Les

unités de calcul sont conçues pour imiter les neurones et la mémoire imite les synapses (Fig. 1.6b). C'est ce qui est utilisé actuellement dans les réseaux de neurones (Fig. 1.6a), brique de base de l'intelligence artificielle. Le calcul neuromorphique utilise également des techniques d'apprentissage inspirées de celles utilisées par le cerveau humain [65]. Par exemple, il utilise des techniques d'apprentissage par renforcement [66], où le système apprend à partir de ses erreurs, et des techniques d'apprentissage non supervisé [67], où cette fois, il apprend à reconnaître des modèles dans les données sans avoir besoin d'une supervision externe. Pour les architectures, deux nouvelles ont été proposées : le calcul proche mémoire (Near-Memory Computing)[68] et le calcul en mémoire (In-Memory Computing) [5, 69].

Le calcul proche mémoire vise à rapprocher le calcul de la mémoire en plaçant des unités de calcul à proximité ou même à l'intérieur de la mémoire. Cela permet de réduire la distance que les données doivent parcourir, ce qui tend à améliorer les performances et l'efficacité énergétique. Cette approche est utilisée avec les technologies de mémoire actuelles comme la SRAM et DRAM : Samsung propose déjà systèmes basés sur des mémoires à haute bande passante (High Bandwith Memory) pour des applications d'intelligence artificielle [70]. Le calcul en mémoire, quant à lui, est une approche qui vise à effectuer des opérations de calcul directement dans la mémoire, sans avoir à déplacer les données vers le processeur. Cela permet de réduire les retards et la consommation d'énergie associés au déplacement des données, avec seulement les entrées et sorties qui se déplacent. Cependant bien que cette architecture puisse être utilisée avec les mémoires traditionnelles, elle fonctionne mieux avec des mémoires spécialisées comme les mémoires résistives que nous allons voir par la suite.

Pour conclure, l'inspiration du cerveau a permis de renouveler l'architecture de Von-Neumann qui pose un problème de consommation excessive d'énergie dans le transfert des données de la mémoire vers le processeur. L'utilisation d'architectures centrées sur la mémoire va permettre de réduire cette consommation et d'améliorer encore une fois les performances des puces de calcul. Pour cela, l'utilisation de nouvelles technologies mémoire est aussi à explorer.

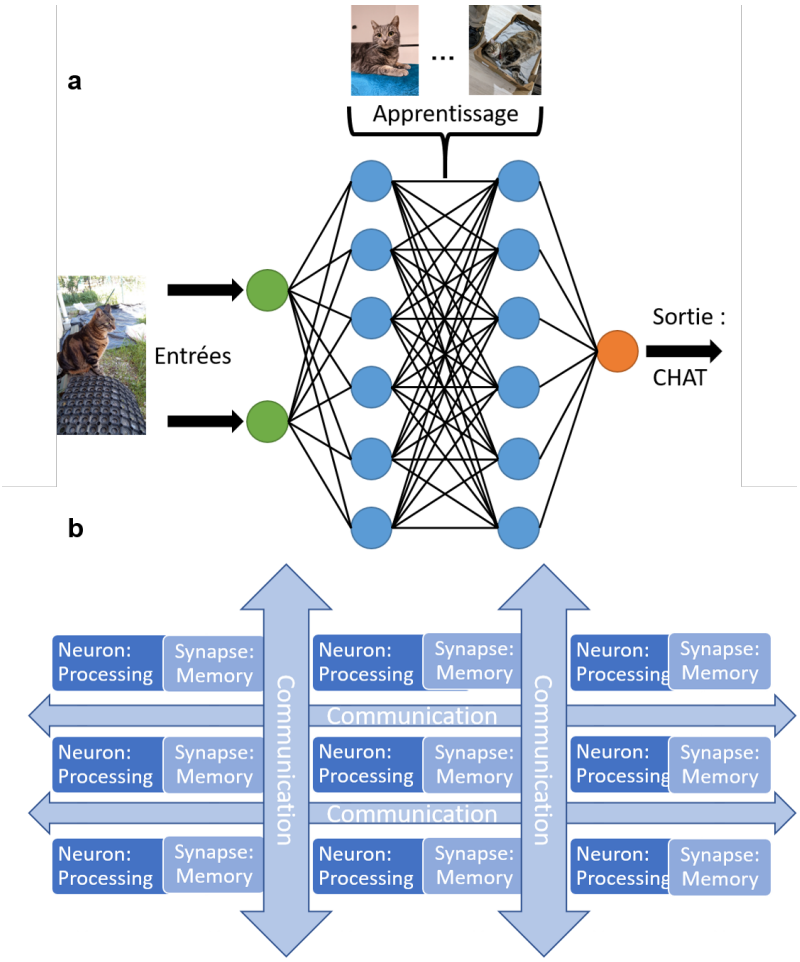


Figure 1.6: **Applications de l'inspiration du cerveau. a** Réseau de neurones pour identifier un chat et **b** Architecture pour le calcul proche mémoire où les neurones (unités de calcul) sont co-localisés avec les synapses (mémoire)

1.2 ... A une nouvelle technologie de mémoire : les memristors

Dans cette section, nous allons explorer l'évolution des technologies de mémoire, en commençant par les mémoires actuelles, SRAM et DRAM, et en progressant vers une technologie émergente prometteuse pour le calcul proche et en mémoire : les memristors.

1.2.1 Mémoires actuelles : SRAM, DRAM, Flash

La mémoire SRAM, introduite dans les années 1960, est composée de transistors et de bascules qui maintiennent un état binaire. Chaque cellule SRAM est constituée de six transistors qui forment deux inverseurs montés en bascule (Fig. 1.7a), stockant ainsi un bit de données. L'état de la cellule est maintenu tant que l'alimentation est présente, ce qui rend la SRAM volatile.

Sa rapidité et sa capacité à conserver des données sans besoin de rafraîchissement la rendent idéale pour des applications nécessitant une grande vitesse, comme la mémoire cache dans les processeurs.

La mémoire DRAM, apparue une décennie plus tard, stocke chaque bit de données dans une cellule composée d'un transistor et d'un condensateur (Fig. 1.7b). Le bit est stocké sous forme de charge dans le condensateur et le transistor agit comme un commutateur qui contrôle l'accès à la charge. Cependant, les condensateurs utilisés dans la DRAM perdent leur charge avec le temps, nécessitant un rafraîchissement régulier des données, ce qui consomme de l'énergie et réduit la vitesse de la mémoire.

La mémoire Flash, apparue dans les années 1980, est une mémoire non volatile qui conserve les données même lorsque l'alimentation est coupée. Elle stocke les données en modifiant les caractéristiques électriques d'une grille flottante dans un transistor (Fig. 1.7c). La présence ou l'absence de charge sur la grille flottante représente un bit de données. Cependant, la mémoire Flash présente des temps d'accès plus lents, en particulier pour les opérations d'écriture et d'effacement, et une durée de vie limitée en termes de nombre de cycles d'écriture.

Malgré leurs avantages, ces technologies de mémoire présentent des contraintes significatives lorsqu'elles sont utilisées pour le calcul en mémoire ou proche de la mémoire. La SRAM, par exemple, a une consommation d'énergie statique importante, principalement due au courant de fuite, même en l'absence d'accès actif. De plus, la nécessité de six transistors par cellule entraîne une taille de cellule relativement grande, impactant la densité de la mémoire et limitant la quantité de mémoire qui peut être intégrée sur une puce.

La DRAM, bien qu'offrant une densité supérieure et un coût inférieur par rapport à la SRAM, nécessite des cycles de rafraîchissement périodiques pour maintenir les données stockées, ce qui consomme de l'énergie et réduit la bande passante mémoire effective disponible pour le calcul. De plus, l'intégration des cellules DRAM avec les unités de traitement dans une architecture de calcul en mémoire est complexe en raison des différences inhérentes à leurs processus de fabrication.

Enfin, bien que la mémoire flash présente de nombreux avantages pour le calcul en mémoire ou proche de la mémoire, elle est intrinsèquement plus lente que la SRAM ou la DRAM, en particulier pour les opérations d'écriture et d'effacement. De plus, elle est sujette à des taux d'erreur plus élevés, en particulier lorsqu'elle est programmée en états multiples, ce qui affecte la fiabilité et nécessite l'implémentation de mécanismes de correction d'erreurs complexes. Enfin, elle a également besoin de tensions de programmation élevées (supérieures à 10V) qui rendent son utilisation compliquée dans les systèmes numériques fonctionnant à plus basse tension.

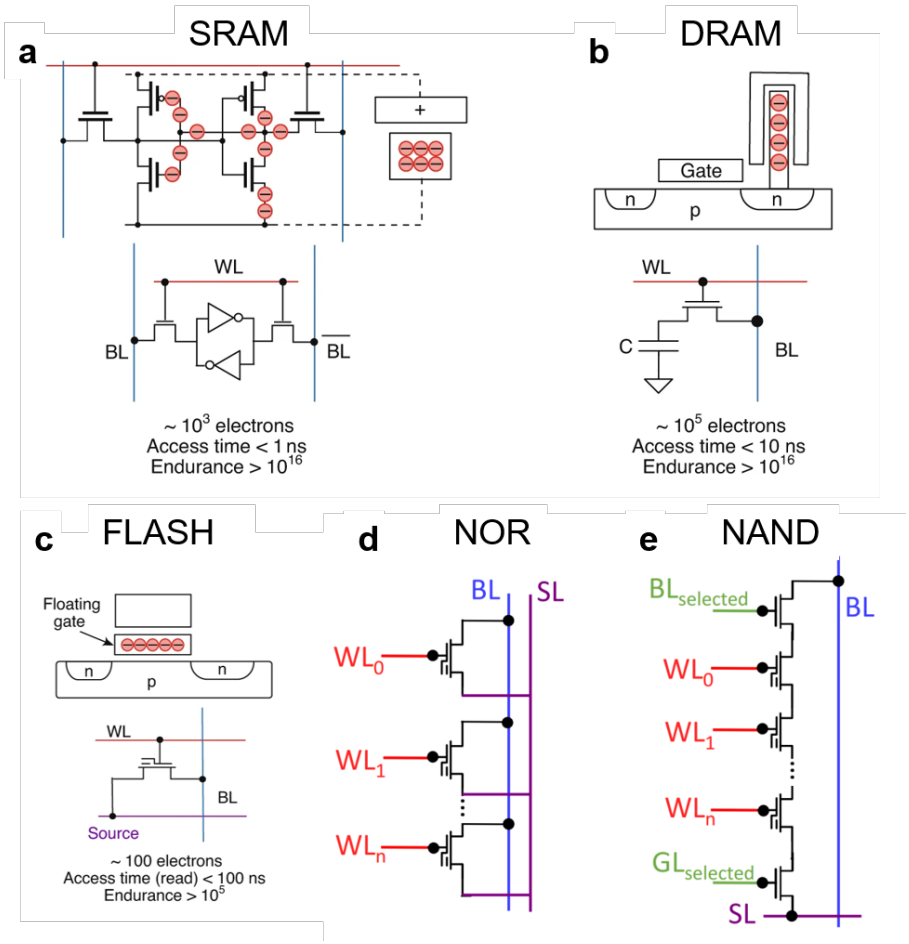


Figure 1.7: **Différentes technologies de mémoire. a** Mémoire SRAM composée de 6 transistors formant deux inverseurs tête-bêche **b** Mémoire DRAM comprenant un condensateur C réalisant le stockage, connecté en série avec un transistor FET **c** Mémoire FLASH, le stockage est ici effectué par une porte flottante d'un transistor FET et peut être utilisé pour **d** la structure NOR ou **e** la structure NAND, tiré de [4]

1.2.2 Différents types de memristors

Le principe dans la recherche des nouvelles technologies de mémoires et d'associer la non volatilité avec la rapidité de la mémoire RAM. Ces nouveaux dispositifs passifs se basent sur des changements d'état de résistance, qui ont été théorisés par Leon Chua [71] et qui ont été découverts en 2008 [72]. Ces mémoires peuvent être de plusieurs catégories : mémoires magnétiques (MRAM), mémoires résistives (RRAM), mémoires à changement de phase (PCM), mémoires et transistors ferroélectriques (FeRAM et FeFET). Cependant, toutes possèdent des avantages par rapport aux mémoires traditionnelles : une transition plus rapide, la possibilité de réaliser des systèmes avec une consommation d'énergie réduite [73] et pour certaines, un comportement analogique qui permettent une utilisation sur plusieurs niveaux.

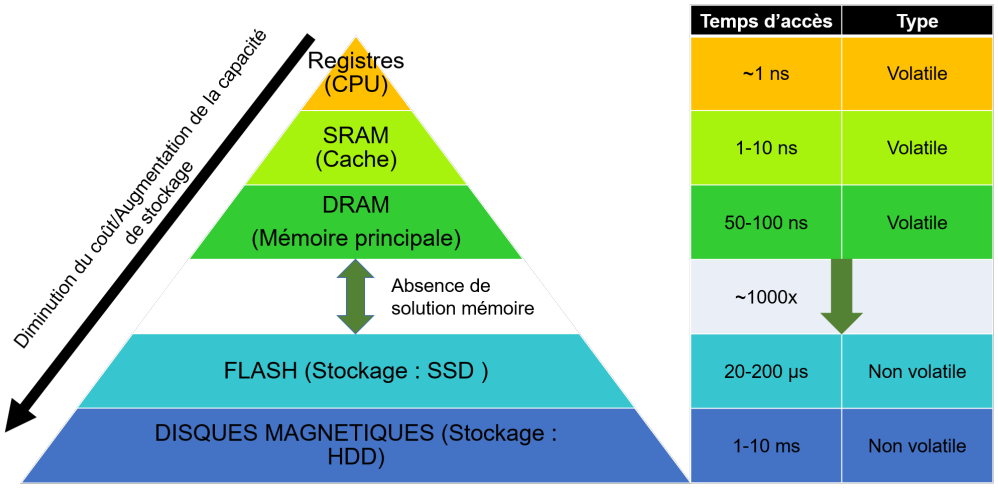


Figure 1.8: **Hiérarchie de la mémoire et comblement du trou avec les nouvelles technologies de mémoires**

STT-MRAM La mémoire STT-MRAM (Spin-Transfer Torque Magnetic Random Access Memory) est une technologie non volatile qui utilise le transfert de spin pour stocker des données. Les données sont stockées sous forme d'orientations magnétiques dans une jonction magnétique à effet tunnel (MTJ) qui est généralement constituée de deux couches de matériaux ferromagnétiques séparées par une fine couche d'isolant [74]. L'application d'un courant à la MTJ peut changer l'orientation magnétique, permettant ainsi l'écriture de données. La STT-MRAM offre une vitesse d'accès comparable à celle de la DRAM, une endurance élevée et une consommation d'énergie relativement faible. Cependant, la complexité de la technologie de transfert de spin peut entraîner des problèmes de fiabilité et de variabilité. De plus, la densité de la STT-MRAM est actuellement inférieure à celle de la DRAM ou de la mémoire flash. Cette technologie est déjà utilisée dans des produits commerciaux comme le processeur Apollo4 qui est utilisé dans les bracelets connectés Fitbit.

RRAM La mémoire RRAM (Resistive Random Access Memory) est une autre technologie non volatile qui utilise des matériaux dont la résistance peut être modifiée par l'application d'un courant électrique[75]. Le fonctionnement de la RRAM repose sur le changement de résistance d'un matériau diélectrique, généralement un oxyde métallique. La RRAM offre une faible consommation d'énergie, une haute densité de stockage et une rapidité d'accès. Cependant, la variabilité du processus de formation et de rupture du filament peut affecter la fiabilité et la prévisibilité de la mémoire. De plus, la durée de vie de la RRAM peut être limitée par le nombre de cycles d'écriture/effacement que le matériau peut supporter avant de se dégrader. Dans la suite du manuscrit, je vais utiliser les termes RRAM et memristors pour parler de ces dispositifs.

PCM La mémoire à changement de phase (PCM) exploite le comportement unique du verre de chalcogénure, un matériau qui peut être commuté entre des états amorphes et cristallins

par l'application de chaleur[76]. Les deux états représentent les valeurs binaires '0' et '1', permettant ainsi le stockage de données. La PCM offre des temps d'accès en lecture rapides, une grande densité de stockage et une durabilité. Cependant, le besoin d'une quantité d'énergie relativement élevée pour changer l'état du matériau de chalcogénure et la dégradation du matériau après un grand nombre de cycles d'écriture peuvent limiter sa durabilité.

FeRAM-FeFET Enfin, les mémoires FeRAM (Ferroelectric Random Access Memory) et les transistors FeFET (Ferroelectric Field-Effect Transistor) sont des technologies non volatiles qui utilisent la ferroélectricité pour stocker des informations[77]. La FeRAM offre des avantages tels que des temps d'accès rapides, une faible consommation d'énergie et une endurance élevée. Cependant, sa densité de stockage est relativement faible par rapport à d'autres technologies de mémoire non-volatile. Les transistors FeFET, qui utilisent un matériau ferroélectrique comme diélectrique de grille dans un FET, offrent des avantages tels que la non-volatilité, la faible consommation d'énergie, et la possibilité d'intégration avec les processus CMOS existants. Cependant, ils présentent également des défis, notamment la difficulté de fabrication et la nécessité de matériaux ferroélectriques de haute qualité pour assurer une performance et une fiabilité optimales.

La RRAM se révèle être dans notre cas, une option économique et accessible pour le calcul en mémoire. Sa compatibilité avec le processus d'intégration back-end CMOS et son comportement analogique la rendent idéale pour des applications comme les réseaux de neurones artificiels. Sa maturité technologique et sa capacité à effectuer des opérations de sampling, telles que l'algorithme de Monte-Carlo (MCMC), la positionnent comme une technologie de mémoire prometteuse pour l'avenir.

1.2.3 Leur intégration pour le calcul en mémoire

L'intégration des memristors dans le calcul en mémoire représente une avancée significative dans le domaine de l'informatique, en particulier pour les applications d'intelligence artificielle. Les memristors, avec leur capacité à maintenir plusieurs états de résistance, offrent une série d'attributs qui les rendent particulièrement adaptés aux systèmes d'IA économes en énergie[73, 78]. Cette capacité leur permet de réaliser des opérations complexes et, surtout, d'approcher les fonctions du cerveau humain, un élément essentiel dans le développement du calcul neuromorphique.

L'intégration de ces dispositifs dans un coeur CMOS est aussi une des question essentielle qui se pose. Plusieurs solutions sont possibles et ont chacune leurs avantages et inconvénients [79] : la première consiste à associer un transistor d'accès pour les adresser séquentiellement (structure 1T1R), une approche similaire à celle utilisée pour les cellules SRAM, on peut donc grâce à ce transistor, programmer et lire le dispositif en appliquant différentes tensions sur la ligne de mots (WL) et la ligne de bits (BL) comme le montre la Fig. 1.10a. Cependant, cette méthode présente des inconvénients, notamment le fait qu'elle nécessite l'utilisation

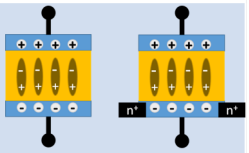
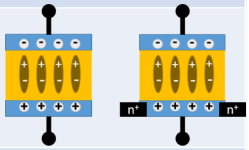
	STT-MRAM	RRAM	PCM	FeRAM | FeFET
Basse résistance « 1 »				
Haute résistance « 0 »				
Principe physique	Une couche magnétique libre est contrôlée par des électrons à spin-polarisés injectés qui traversent une couche magnétique fixée par un couple de transfert de spin	Formation et destruction d'un filament conducteur	Transition de phase entre une phase cristalline (basse résistivité) et une phase amorphe (haute résistivité)	Les matériaux ferroélectriques ont une polarisation électrique spontanée qui peut être inversée avec un champ électrique externe
Avantages	- Technologie mature - Très rapide - Très bonne endurance - Résistante aux radiations - Compatible Back-end CMOS	- Technologie mature - Bonne rétention même à haute température - Structure simple - Rapidité et faible courant de commutation - Compatible Back-end CMOS	- Grand ratio basse/haute résistance - Bonne endurance - Miniaturisation possible - Compatible Back-end CMOS - Rapidité de commutation	- Rapidité de commutation - Haute densité - Bonne endurance - Utilisation des FeFET comme de la DRAM : compatible avec des transistor haute performance - Opportunité avec les jonctions tunnel ferroélectrique
Inconvénients	- Fort courant de commutation - Faible ratio basse/haute résistance - La miniaturisation affecte la capacité mémoire - Compromis entre rapidité et fiabilité	- Endurance faible - Haute variabilité (cyclique et entre dispositifs) - Besoin d'une étape de Formation initiale - Fiabilité de la commutation	- RESET très énergivore - Faible rétention au haute température - Dérive de la résistance avec le temps	- Lecture destructrice - Faible densité - Faible endurance des FeFET (comme de la SRAM)

Figure 1.9: **Structures et récapitulatif des nouvelles technologies mémoire**

d'un transistor qui est en dessous du premier niveau métallique pour adresser des dispositifs qui peuvent être aussi hauts que le niveau métallique 4-5, diminuant ainsi la densité de mémoire. Pour l'augmenter, une autre solution consiste à uniquement placer un dispositif. Pour le programmer ou le lire, il faut juste appliquer une différence de tension entre WL et BL. Le problème qui se pose dans cette configuration est que toutes les autres cellules recevront la moitié de cette différence de tension, ce qui induira des courants de fuite qui circuleront à travers tous les dispositifs à moitié sélectionnés (Fig.1.10b). La dernière méthode est d'associer une diode bipolaire qui arrête le courant positif et négatif faible, ce qui élimine les effets des chemins de fuite dans une grande partie de la matrice de mémoire mais pas la totalité (Fig..1.10c).

Maintenant que les dispositifs sont intégrés, il est essentiel de développer des architectures qui permettent d'exécuter efficacement les opérations fondamentales des réseaux de neurones. Ces opérations consistent principalement en des multiplications et des accumulations successives, communément appelées opérations Multiply-and-ACcumulate (MAC). Une structure particulièrement prometteuse pour réaliser ces opérations est la structure en réseau, ou "crossbar". Cette structure est une grille bidimensionnelle où les dispositifs de mémoire,

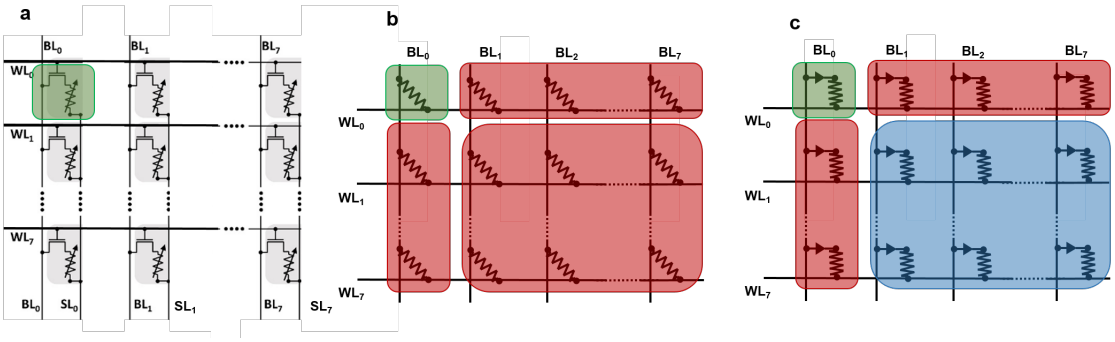


Figure 1.10: **Structures d'utilisation des nouvelles technologies mémoire.** **a** 1T1R, cellule composée d'un transistor et d'un memristor, **b** cellule composée uniquement d'un memristor, l'application d'une différence de tension entre BL et WL sur la cellule verte (celle programmée) se répercute sur l'ensemble des autres memristors, en rouge, et **c** cellule composée d'un sélectionneur et d'un memristor (1S1R), seuls les cellules en rouge reçoivent la différence de tension, celles en bleu sont intactes.

sont placés aux intersections des lignes et des colonnes (Fig1.10). Cette configuration permet une densité élevée de dispositifs de mémoire et offre la possibilité d'effectuer des opérations de calcul en parallèle directement dans la mémoire.

$$I_i = \sum_j G_{i,j} V_j \tag{1.1}$$

La première architecture en réseau utilise les lois d'Ohm et de Kirchhoff et l'utilisation des memristors de façon analogique. Dans cette architecture, la conductance du dispositif représente le poids synaptique du réseau de neurones, les entrées sont les tensions appliquées aux lignes et les sorties sont lues avec les courants des colonnes (Fig. 1.11a-b). Il est donc possible de réaliser une multiplication matrice-vecteur avec la formule suivante :

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} G_{1,1} & G_{1,2} & G_{1,3} & \cdots & G_{1,m} \\ G_{2,1} & G_{2,2} & G_{2,3} & \cdots & G_{2,m} \\ G_{3,1} & G_{3,2} & G_{3,3} & \cdots & G_{3,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ G_{n,1} & G_{n,2} & G_{n,3} & \cdots & G_{n,m} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_m \end{bmatrix} \tag{1.2}$$

En effectuant l'opération de multiplication et d'accumulation directement au sein du réseau, le calcul analogique en mémoire peut réduire de manière significative le mouvement de données entre la mémoire et les unités de calcul, qui est une source majeure de consommation d'énergie dans les architectures traditionnelles. De plus, le parallélisme inhérent à la structure en réseau permet un haut degré de simultanéité, pouvant potentiellement conduire à des améliorations substantielles du débit de calcul. En tirant parti des niveaux continus de conductance des dispositifs pour représenter des poids synaptiques multi-niveaux, on offre la possibil-

ité de calculs de précision supérieure par rapport aux réseaux neuronaux à poids binaires. Cela peut conduire à une meilleure précision dans les tâches d'inférence sans encourir la surcharge d'énergie et de surface généralement associée à un calcul numérique de précision supérieure. Des recherches en utilisant cette application ont été récemment réalisées [5, 78, 80–84].

Dans cet article [84], l'auteur présente une puce fabriquée avec un processus hybride CMOS/RRAM en 40 nm composée de 4096 cellules 1T1R RRAM pour effectuer du calcul en mémoire. L'efficacité enégétique moyenne de cette puce est estimée allant de 9.81 TOPS/W en mode 8 bits à 75.17 TOPS/W en mode 64 bits pendant le fonctionnement à une tension d'alimentation 900 mV et une fréquence de 75 MHz.

Cependant, l'application concrète des réseaux utilisant des memristors dans les produits industriels requiert une analyse détaillée de tous les éléments indispensables à la fiabilité du système. En effet, les défis résident notamment dans les imperfections des dispositifs. La maîtrise précise de la résistance de chaque memristor, qui représente un poids synaptique dans les réseaux neuronaux, est actuellement limitée par la variabilité de ces dispositifs [5, 13, 73]. Cette imprécision peut entraîner une résolution limitée des dispositifs, des circuits périphériques d'écriture lourds et précis, ou encore l'utilisation de systèmes correcteurs d'erreur complexes, le tout ayant un impact majeur sur la consommation du système. Au niveau du circuit, une complexité de conception plus élevée (comme les opérations MAC sont effectuées en analogique et les autres opérations sont en numérique) nécessite la conversion entre les signaux analogiques et numériques, donc, des circuits de convertisseurs analogiques-numériques (CAN) et numériques-analogiques (CNA) complexes sont requis, avec des dispositifs de haute précision, haute vitesse et tolérants au bruit.

Une approche différente développée par les chercheurs de Samsung Advanced Institute of Technology et Samsung Electronics [6], consiste à utiliser une matrice qui tire parti de la sommation de résistance pour les opérations analogiques de multiplication-accumulation, en lieu et place de la sommation de courant. Cette dernière fait, en effet, face à des contraintes dimensionnelles découlant du fort courant associé à de nombreux résistances en parallèle, en particulier dans le cas d'états de faible résistance. Ce courant élevé peut entraîner une augmentation de la taille des circuits périphériques et un impact accru des résistances parasites dans les métaux de routage. Cette nouvelle architecture repose sur une matrice composée de MRAM (Fig.1.12a), sélectionnée pour ses états de faible résistance. Elle a été utilisée pour créer un modèle de réseau neuronal binaire pour la classification d'images. Elle utilise deux chemins parallèles, chacun avec une MRAM et un MOSFET en série (Fig.1.12b). Les résistances de poids synaptiques sont stockées dans chaque chemin, l'un contenant le poids et l'autre son complément. Le choix du chemin, déterminé par la tension d'entrée, permet une opération XNOR analogique (Fig.1.12d), correspondant à l'opération de multiplication du réseau neuronal binaire. La résistance de chaque colonne du tableau, obtenue par la somme des résistances de sortie des cellules binaires, remplace la somme de courant des matrices de points de croisement traditionnelles. Cette résistance est lue via une méthode de domaine

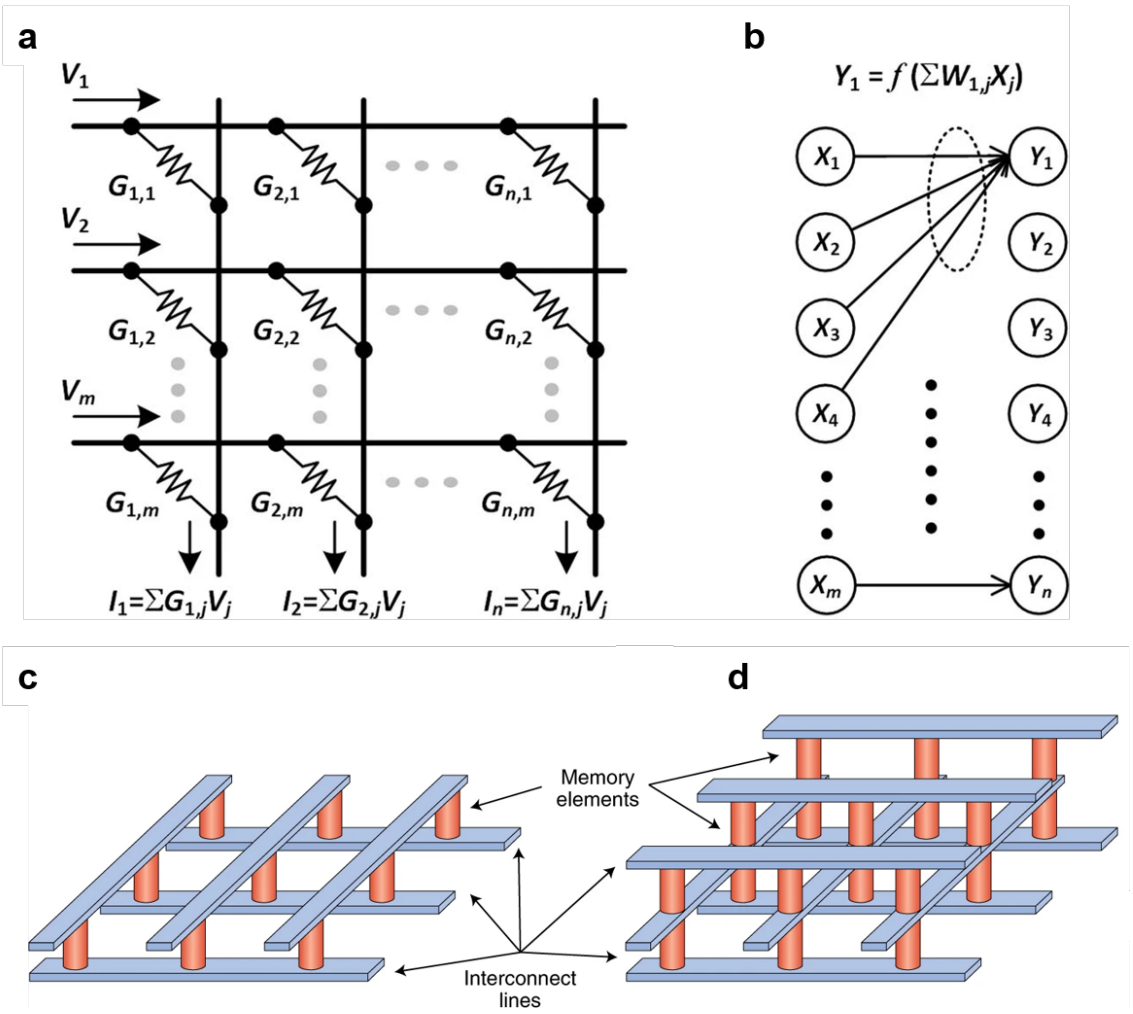


Figure 1.11: **Architecture crossbar pour le calcul en mémoire analogique a** Schéma de l'architecture crossbar pour réaliser les opérations MAC d'un **b** réseau de neurones, **c** structure croisée avec composée de RRAM sur une seule couche et **d** sur plusieurs couches (tiré de [5] et [6])

temporel (Fig.1.12c), capturée par un convertisseur temps-numérique (TDC). Malgré la nature binaire de la MRAM, cette architecture promet une consommation d'énergie réduite grâce à l'utilisation de courants de charge et de décharge pour les calculs. Pour surmonter cette limitation, l'intégration de memristors pourrait être envisagée. Ces derniers, capables de stocker une gamme plus large d'états de résistance, permettraient d'améliorer la précision des valeurs de poids stockées et d'augmenter la précision des calculs dans le modèle de réseau neuronal. Dans cette étude, les chercheurs présentent deux crossbars de 64 x 64 MRAM fonctionnant à 11.1 MHz et prévoient une efficacité moyenne de 405 TOPS/W pour une tension d'alimentation de 0.8 V et 262 TOPS/W pour 1.0 V.

En conclusion, l'intégration des memristors pour le calcul en mémoire offre des perspectives prometteuses pour l'avenir de l'IA et d'autres applications nécessitant une grande capac-

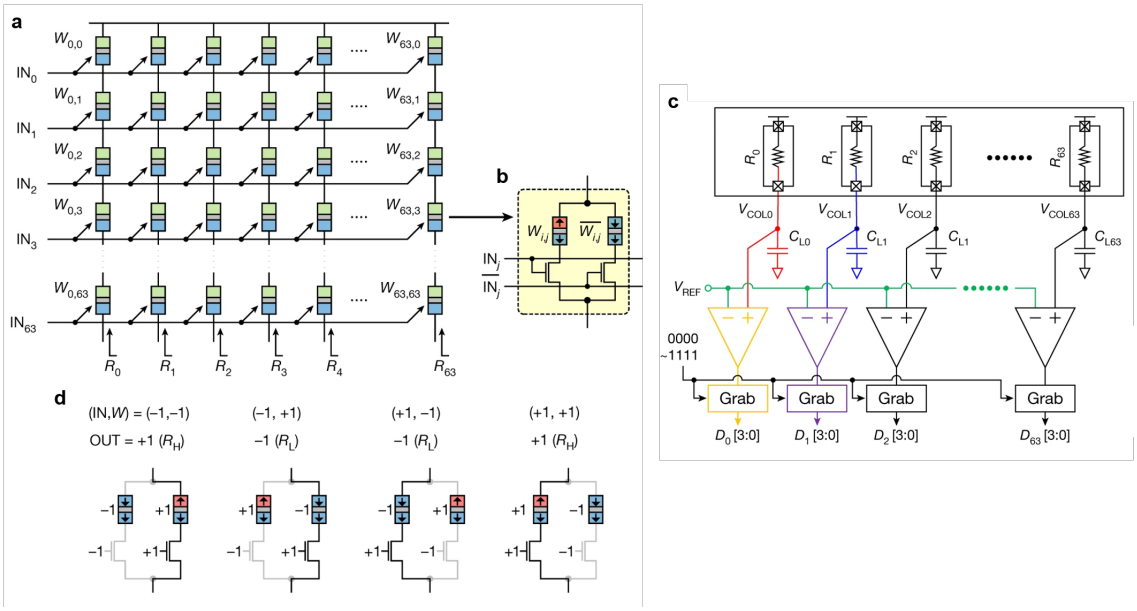


Figure 1.12: **Architecture crossbar pour le calcul en mémoire avec sommation des résistances a b c d** (tiré de [6])

ité de stockage et de traitement des données. Cependant, des recherches supplémentaires et des avancées technologiques sont nécessaires pour surmonter les défis associés à leur mise en œuvre pratique. Les avancées dans la technologie des memristors, les techniques de conception de circuits, les architectures au niveau du système et le niveau algorithmique, comme le développement de modèles d'IA adaptés au matériel, seront essentielles pour réaliser pleinement le potentiel des memristors dans le calcul en mémoire.

Dans la suite de la thèse, nous allons explorer un autre domaine de l'intelligence artificielle et de l'architecture neuromorphique, le raisonnement bayésien. Le raisonnement bayésien et les réseaux de neurones sont deux approches distinctes de l'apprentissage automatique, chacune avec ses propres avantages et inconvénients. Le raisonnement bayésien est basé sur le théorème de Bayes, qui permet de mettre à jour la probabilité d'une hypothèse en fonction de nouvelles données. Il est particulièrement utile lorsque les données sont limitées ou incertaines, permettant d'incorporer des connaissances a priori dans le modèle. Les réseaux de neurones, en revanche, sont inspirés par le fonctionnement du cerveau humain et sont composés d'unités de traitement (neurones) organisées en couches. Ils sont particulièrement efficaces pour apprendre des représentations complexes à partir de grandes quantités de données, notamment dans des domaines tels que la reconnaissance d'images et le traitement du langage naturel.

Les réseaux de neurones peuvent modéliser des fonctions très complexes et s'adapter à une grande variété de tâches, mais ils nécessitent souvent de grandes quantités de données et de puissance de calcul, et leurs mécanismes internes peuvent être difficiles à interpréter. Le raisonnement bayésien offre une interprétabilité et une compréhension claires des mod-

èles, permettant une quantification explicite de l'incertitude. Je vais donc présenter dans ces travaux comment intégrer ce raisonnement dans des puces exploitant l'architecture proche mémoire avec des memristors pour réaliser des applications plus ou moins complexes avec un coût énergétique faible.

Chapter 2

La communauté de la Machine Bayésienne

Le modèle de fonctionnement d'une machine Bayésienne

> Common sense is indeed sufficient to show us that, from the observation of what has in former instances been the consequence of a certain cause
>
> — An Essay towards Solving a Problem in the Doctrine of Chances. By the Late Rev. Mr. Bayes, Communicated by Mr. Price, in a Letter to John Canton

DÉFINIR L'INTELLIGENCE est une tâche complexe. Cependant, il est indéniable qu'une part significative de l'intelligence réside dans notre capacité à identifier nos propres lacunes en matière de connaissance. Pour de nombreux systèmes dits intelligents, comme les robots, les véhicules autonomes ou les diagnostics médicaux, la prise en compte de l'incertitude est essentielle. Cette incertitude, inhérente à l'interaction avec des données limitées, des capteurs bruités ou des systèmes critiques en termes de sécurité, doit être correctement modélisée. Pour les adeptes de l'approche bayésienne, le langage mathématique universel pour exprimer l'incertitude est celui des lois de probabilité. Tout comme pour les êtres humains, le monde dans lequel une intelligence artificielle évolue est empreint d'incertitude. Les données qu'elle rencontre peuvent être bruitées, partielles ou biaisées. L'intégration de la théorie des probabilités dans la conception de systèmes se présentant comme des machines intelligentes est donc une perspective très prometteuse.

Au cours des dernières années, les progrès de l'intelligence artificielle ont été largement portés par les avancées des réseaux neuronaux artificiels. Cependant, même le système d'intelligence artificielle le plus sophistiqué au monde aura moins de bon sens qu'un animal. Les animaux ont la capacité d'apprendre à partir de très peu d'exemples, ce qui est un défi majeur pour les réseaux neuronaux artificiels. Par exemple, un enfant peut reconnaître un chat après avoir vu un seul dessin de chat. Aucune intelligence artificielle n'est actuellement capable d'un tel exploit. Pour identifier un chat, un réseau neuronal doit être entraîné sur des données de chats, c'est-à-dire qu'il doit être exposé à des milliers d'images de chats, mélangées à d'autres données, pour pouvoir distinguer si une image représente un chat ou non. De plus, la sortie d'un réseau neuronal produira des réponses presque certaines, alors que lorsqu'un enfant demande une confirmation, il peut n'être sûr de sa prédiction qu'à 50%, illustrant ainsi son incertitude.

Dans ce chapitre, adapté de l'article [85], je démarre par introduire les principes de l'inférence bayésienne notamment les simplifications possibles dues à l'indépendance conditionnelle et les méthodes de multiplications possibles pour l'utilisation dans un circuit logique économe en énergie. Je continue par l'analyse de l'architecture de la machine bayésienne et par une étude approfondie sur le choix des valeurs des seeds pour le calcul stochastique. Je termine par une application pratique de la machine bayésienne, la reconnaissance de gestes, du fonctionnement aux résultats en passant par l'analyse avec des faibles datasets et l'explicabilité de la machine bayésienne.

2.1 Principes Fondamentaux de l'Approche Bayésienne

Le théorème de Bayes est la base de toutes les techniques d'inférence bayésienne. Il fournit une méthode pour mettre à jour les probabilités en fonction de nouvelles données. Mathématiquement, le théorème de Bayes est exprimé comme suit :

$$p(A|B) = \frac{p(B|A) \times p(A)}{p(B)} \tag{2.1}$$

où :

- $p(A|B)$ est la vraissemblance (ou likelihood en anglais) a posteriori de A étant donné B
- $p(B|A)$ est la vraissemblance de B étant donné A
- $p(A)$ est la probabilité a priori de l'événement A
- $p(B)$ est la probabilité a priori de l'événement B l'on peut très souvent simplifier et ne garder que l'équation suivante :

$$p(A|B) \propto p(B|A) \times p(A) \tag{2.2}$$

Ce processus de mise à jour des probabilités nous permet de réviser nos croyances sur une hypothèse (évenement A) à mesure que nous recevons de nouvelles informations (évenement B). De plus, curieusement, il reflète la façon dont les humains apprennent et comprennent de nouveaux concepts [86, 87]. Lorsque les humains rencontrent de nouvelles informations, ils mettent à jour leurs croyances pour refléter ces nouvelles connaissances. Ce processus qui se met à jour basé sur de nouvelles informations est un aspect crucial du processus d'apprentissage et est analogue à la mise à jour bayésienne des probabilités en utilisant le théorème de Bayes.

2.1.1 Indépendance Conditionnelle : Au-delà de l'Approximation Naïve

L'inférence bayésienne permet d'évaluer la probabilité d'un événement Y, par exemple, l'occurrence d'une urgence médicale, sur la base d'un ensemble d'observations. Par exemple, une valeur de un pour Y pourrait représenter l'occurrence d'une crise mineure, une valeur de deux une crise majeure, et une valeur de zéro l'absence de toute crise.

La loi de Bayes fournit alors la probabilité que ces situations se produisent actuellement, en fonction d'une probabilité de se produire à tout moment (le prior $p(Y = y)$) et des facteurs de vraisemblance $p(O_1, O_2, ..., O_n|Y = y)$ qui modélisent le comportement des capteurs O_1, O_2,...,O_n en l'absence ou en présence d'une crise :

$$p(Y = y|O_1, O_2, ..., O_n) \propto p(O_1, O_2, ..., O_n|Y = y) \times p(Y = y). \tag{2.3}$$

Les facteurs de vraisemblance présentent un coût de mémoire prohibitif, qui croît exponentiellement avec le nombre d'observations n. Dans des situations réelles, ce coût peut être allégé. Dans notre exemple, les capteurs qui mesurent des aspects distincts des patients (par exemple, le rythme cardiaque et la température corporelle) peuvent souvent être considérés comme conditionnellement indépendants (ce qui signifie que, une fois que l'on sait qu'une crise est en cours, les valeurs du rythme cardiaque et de la température corporelle peuvent être considérées comme des processus statistiquement indépendants). Si toutes les observations sont conditionnellement indépendantes, l'équation 2.3 se simplifie en

$$p(Y=y|O_1,O_2,...,O_n) \propto p(O_1|Y=y) \times p(O_2|Y=y) \times p(O_3|Y=y)... \times p(O_n|Y=y) \times p(Y=y), \tag{2.4}$$

avec un coût de mémoire pour la vraisemblance qui croît maintenant linéairement avec n, et des facteurs de vraisemblance qui deviennent faciles à modéliser sur la base des mesures des valeurs des capteurs, par exemple, pendant l'occurrence ou en l'absence d'une crise.

Si on transpose l'équation 2.4 en terme d'architecture, nous obtenons la figure 2.2a qui se place très bien dans le concept de calcul proche mémoire. En effet, les matrices mémoires où sont stockées les vraisemblances sont dans le même bloc, juste à coté des multiplicateurs. Cette architecture va nous permettre de réaliser l'inférence bayésienne avec le minimum possible de mouvements de données et donc de réduire les coûts énergétiques comme nous le verrons dans le chapitre suivant.

Les résultats présentés dans ce chapitre et le suivant sont considérés dans des cas où toutes les observations peuvent être considérées comme conditionnellement indépendantes étant donné la variable inférée Y (inférence bayésienne naïve). Ce n'est pas toujours le cas. En particulier, dans une situation où deux capteurs redondants mesurent le même phénomène, ils ne peuvent jamais être considérés comme indépendants dans un bon modèle, même conditionnellement à la variable inférée. Par exemple, si les observations O_2 et O_3 ne sont pas conditionnellement indépendantes, la vraisemblance $p(O_2,O_3|Y=y)$ ne peut pas être factorisée, et un modèle d'inférence bayésienne approprié peut se lire comme suit :

$$p(Y=y|O_1,O_2,...,O_n) \propto p(O_1|Y=y) \times p(O_2,O_3|Y=y) \times = ... \times p(O_n|Y=y) \times p(Y=y). \tag{2.5}$$

Ce modèle peut toujours être mis en œuvre par une machine bayésienne. Dans ce cas, les observations O_2 et O_3 doivent être regroupées en une seule colonne, et leur valeur conjointe doit être utilisée pour adresser les tableaux de vraisemblance de cette colonne (voir Fig. 2.2b). Néanmoins, il convient de souligner que le coût de la mémoire de la machine bayésienne augmente avec le nombre de variables non conditionnellement indépendantes.

2.1.2 Techniques de Multiplication dans l'Analyse Bayésienne

Un défi important de la machine bayésienne dans une intégration dans un circuit physique est que les multiplications sont normalement une opération coûteuse en CMOS, ce qui soulève une préoccupation si un multiplicateur est associé à chaque tableau de mémoire de vraisemblance. Pour cette raison, nous nous appuyons sur le calcul stochastique[88, 89]. Ce paradigme de calcul code les probabilités sous forme de flux de bits aléatoires, où, à chaque cycle d'horloge, la probabilité que le bit soit un est simplement la probabilité codée. La multiplication des probabilités peut alors être réalisée à l'aide de simples portes ET (voir Fig. 2.1a), avec un coût

de surface extrêmement minimal[88]. Un aspect important de ce modèle de calcul est que, comme dans la plupart des situations pratiques, les probabilités ont tendance à être faibles, la sortie des portes ET du calcul stochastique est une valeur zéro à la plupart des cycles d'horloge. Par conséquent, les différents blocs de la machine bayésienne n'ont besoin de passer que des bits simples (Fig. 2.2a) qui sont des zéros à la plupart des cycles d'horloge : la machine bayésienne limite considérablement le mouvement des données. Dans l'ensemble, en raison de cette simplicité, la machine bayésienne ressemble juste à une puce mémoire – nous l'appelons une mémoire "nativement intelligente".

Au-delà de son élégance conceptuelle, la conception de la machine bayésienne a dû faire face à plusieurs défis liés à l'utilisation du calcul stochastique. Premièrement, le calcul stochastique nécessite des générateurs de nombres aléatoires. Notre conception utilise des registres à décalage à rétroaction linéaire (LFSR) pour générer des nombres uniformément répartis de manière aléatoire. Un seul LFSR est utilisé par colonne (Fig. 2.2c) : chaque ligne effectuant un calcul stochastique indépendant, les différentes lignes peuvent se baser sur les mêmes nombres pseudo-aléatoires. Ensuite, à chaque cycle d'horloge, chaque bloc de vraisemblance génère un bit aléatoire avec la probabilité p en comparant le nombre généré par le LFSR vertical avec la valeur de la probabilité p lue à partir du tableau de mémoire de vraisemblance.

Par ailleurs, les vraisemblances dans les modèles bayésiens ont tendance à être très faibles, et le calcul stochastique converge lentement lors de la multiplication de faibles probabilités[90]. En effet, comme la multiplication des probabilités donne un résultat de plus en plus faible, le nombre de 1, résultat des différentes portes ET diminue : c'est ce qu'on appelle la dilution. Ce problème met également en avant deux autres problématiques du calcul stochastique notamment la durée et la précision, si à chaque cycle d'horloge, un seul bit est généré, pour avoir une précision suffisante, il faut faire un grand nombre de cycles d'horloges, donc un temps plus long de calcul qui entraîne également une plus grande consommation énergétique.

Un autre problème majeur du calcul stochastique et que nous allons voir dans la partie 2.2.2, est la corrélation des flux de bits. En effet, pour pouvoir fonctionner correctement, comme le montre la figure 2.1b, les deux flux ne doivent pas êtres semblables, donc non corrélés.

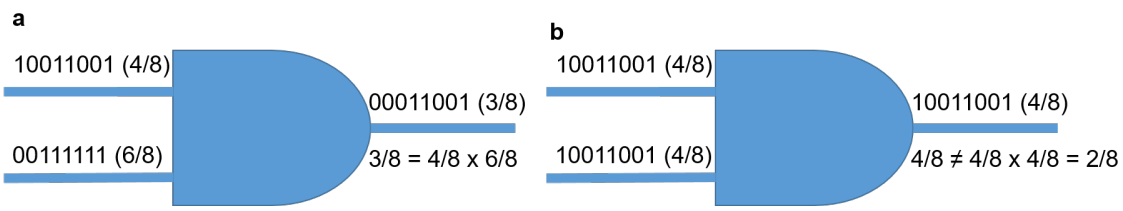


Figure 2.1: **Calcul stochastique pour deux flux de bits. a** opération de multiplication si les flux ne sont pas corrélés, le résultat est bien le produit des deux flux en entrée et **b** si les flux sont corrélés, le résultat ne correspond pas au produit des deux flux en entrée

2.2 Intégration de la Machine Bayésienne dans la Conception des Circuits

2.2.1 Exploration de l'Architecture

Le design complet, se compose d'une cellule de vraisemblance ('likelihood') répétée de nombreuses fois. Comme le montre la Figure 2.2c, chaque cellule de vraisemblance sera donc composée d'une matrice de mémoire, d'un générateur de nombres aléatoires, d'un circuit capable de transformer ces bits aléatoires en un flux de bits proportionnel à la valeur de probabilité lue à partir du tableau de mémoire et d'une porte ET qui effectue le produit de probabilité. De plus, pour pouvoir contrôler chacune de ces cellules, mais aussi pour gérer les entrées et sorties du système, d'autres blocs sont ajoutés. Le contrôleur de mémoire a pour tâche de gérer l'adressage des mémoires pour l'écriture et la lecture. Les contrôleurs d'entrée/sortie sont utilisés pour gérer les données. Le contrôleur d'entrée est utilisé pour gérer l'écriture des mémoires, la génération du flux de bits stochastiques pour le prior, et le chargement des graines nécessaires à la génération de nombres pseudo-aléatoires. Ils sont tous réunis dans le contrôleur numérique de la figure 2.2.

Maintenant que cette architecture est fixée, nous allons pouvoir modifier les multiplicateurs pour les remplacer par des comparateurs entre un nombre aléatoire et la probabilité (vraisemblance) stockée dans la mémoire. Pour cela, comme je l'ai dit précédemment, pour les nombres aléatoires, nous utilisons des générateurs de nombre pseudo-aléatoires en se basant sur des registres à rétro-action linéaire (LFSR). C'est un générateur pseudo-aléatoire qui est économique en termes de superficie et de consommation d'énergie car la mise en œuvre ne nécessite que des bascules et des portes XOR. La structure physique d'un LFSR est souvent présentée sous forme polynomiale, par exemple $x^8 + x^6 + x^5 + x^4 + 1$ pour un LFSR de 8 bits. Ce polynôme correspond aux différents bits du registre à décalage qui sont connectés les uns après les autres par des portes logiques XOR. Pour chaque taille de LFSR, une configuration maximise sa périodicité, passant dans tous les états sauf l'état nul. Cependant, il possède trois inconvénients majeurs pour notre système. Tout d'abord, l'état zéro n'est jamais atteint, ce qui réduit d'une unité le nombre de valeurs atteintes par le générateur. Deuxièmement, ce générateur est pseudo-aléatoire, ce qui signifie que la séquence de bits générés est périodique, ce qui limite la longueur de la configuration du flux de bits de sortie. De plus, le nombre de LFSRs de la même structure physique qui peuvent être mis en série pour le produit stochastique est limité. En effet, deux LFSRs doivent être différents, sinon les flux de bits en sortie des différentes vraisemblances seront fortement corrélés les uns avec les autres. Et troisièmement, pour générer différentes valeurs, chaque LFSR a besoin d'une valeur de seed qui doit être programmée, augmentant ainsi la complexité du système et nécessitant également d'être initialisé à chaque fois que nous éteignons le système.

Pour effectuer la comparaison, il faut que la probabilité stockée et que le nombre aléatoire

généré soient codés sur le même nombre de bits. Ensuite, nous pouvons utiliser un comparateur conventionnel qui va comparer si le nombre aléatoire généré est inférieur ou égal à la probabilité affichée, le bit de sortie est 1, sinon, il est 0. Une autre méthode, proposée par Gupta en 1988 [7] et présentée en Figure 2.3, consiste à générer le flux de bits proportionnel en comparant chaque bit de la probabilité avec le nombre aléatoire généré par le LFSR. Le système est basé sur le principe que chaque bit du LFSR a une probabilité de $1/2$ d'être un 1 et une probabilité de $1/2$ d'être un 0. Les portes ET sont utilisées pour pondérer chaque bit généré par le LFSR par la probabilité chargée dans le registre. La porte OR logique est utilisée pour ajouter ces différentes options. Une étude réalisée par [64] montre que le circuit Gupta est plus performant en énergie et en surface que le comparateur classique.

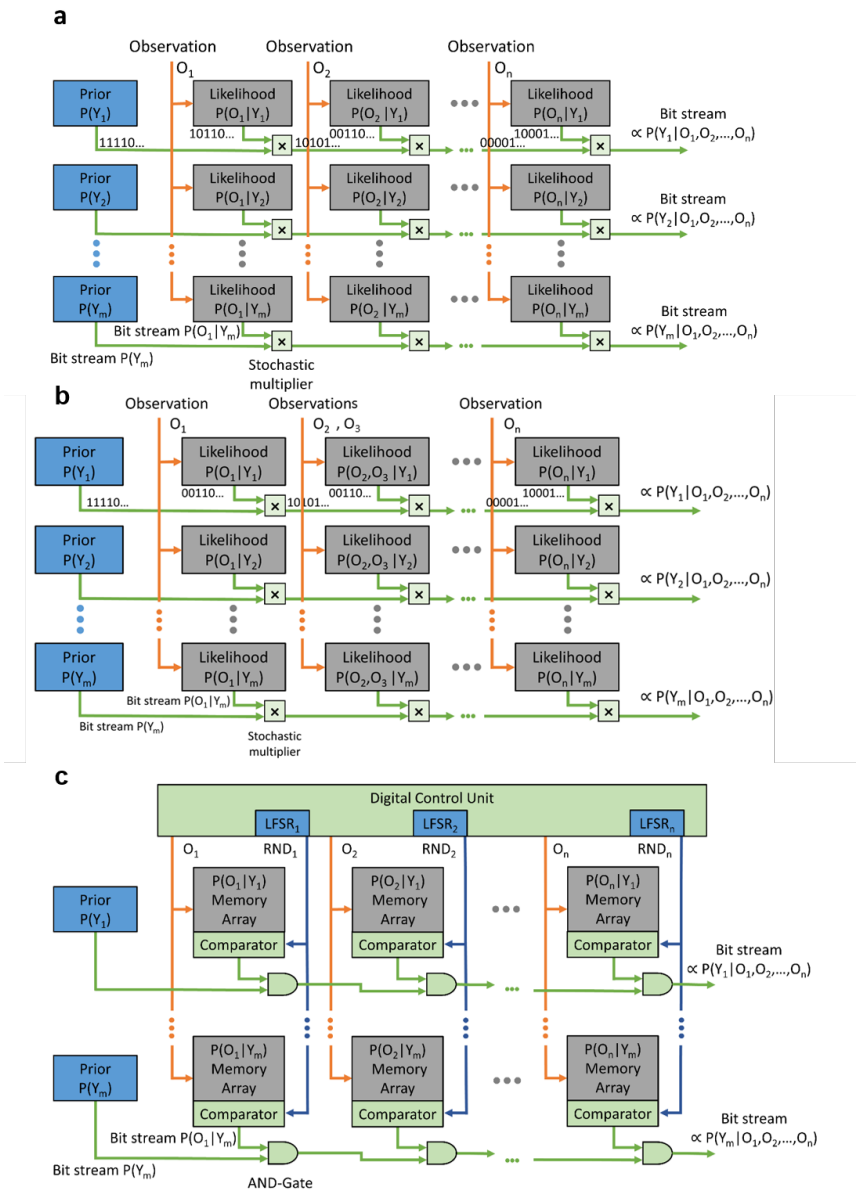


Figure 2.2: **Architecture générale de la machine bayésienne.** **a** Architecture générale. Les vraisemblances sont stockées dans des matrices de mémoire de vraisemblance implémentées par des matrices de memristors. Les observations du monde réel choisissent la valeur de probabilité appropriée à partir des matrices de mémoire de vraisemblance, sur la base desquelles des flux de bits stochastiques proportionnels sont générés et multipliés par un multiplicateur stochastique. En sortie, les flux de bits encodent naturellement la distribution a posteriori. **b** Architecture de la machine bayésienne avec des observations non conditionnellement indépendantes. Cette architecture réalise une inférence bayésienne non naïve suivant l'équation 2.5, en regroupant les observations O_2 et O_3 dans la même colonne. **c** Optimisation de la machine bayésienne pour le matériel. Des nombres aléatoires (RND) sont générés à l'aide de registres à décalage à rétroaction linéaire (LFSRs), partagés par colonne, puis convertis en utilisant des circuits numériques "Gupta" en une série de bits aléatoires proportionnels à la probabilité appropriée. De plus, les vraisemblances sont normalisées par la valeur de vraisemblance maximale de la colonne afin de maximiser la vitesse de convergence de la machine. La multiplication stochastique est mise en œuvre à l'aide d'une porte logique ET à un seul bit.

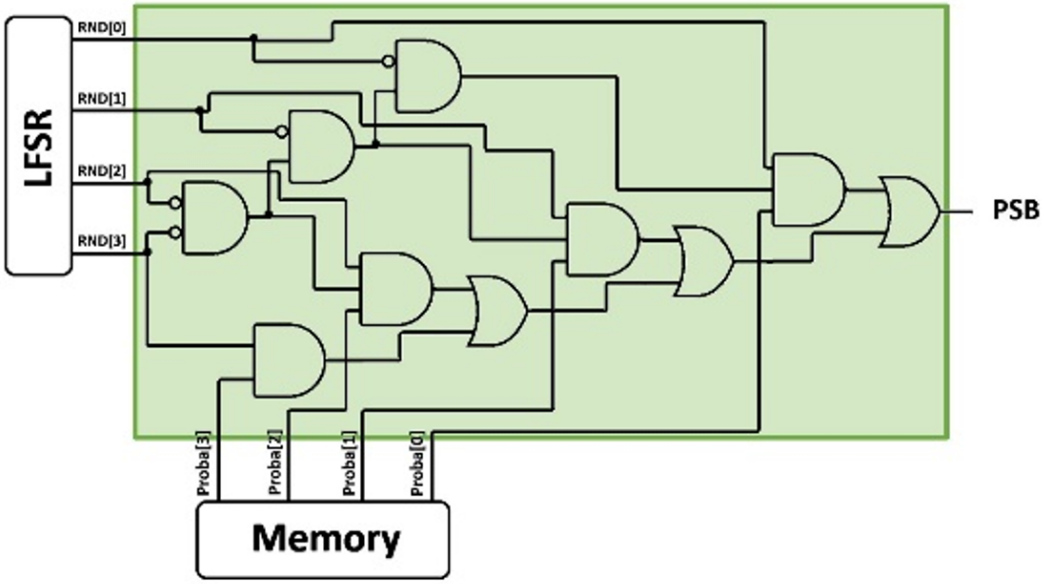


Figure 2.3: **Schéma du circuit comparateur de Gupta** [7], utilisé pour générer le flux de bits aléatoires de vraisemblance en comparant la probabilité lue en mémoire avec le nombre aléatoire généré par le LFSR. Pour des raisons de compacité, le circuit est présenté ici dans une version à quatre bits. Une version à huit bits est mise en œuvre sur la puce.

2.2.2 L'Importance du Choix des Seeds

Dans sa forme originale, le calcul stochastique repose sur des nombres aléatoires de haute qualité et non corrélés. Générer des bits aléatoires de haute qualité et maintenir l'indépendance des bits stochastiques après qu'ils aient été traités par des circuits de calcul stochastique est un défi majeur, conduisant à des stratégies coûteuses comme l'isolation et la régénération de l'aléatoire [90]. Cependant, en pratique, de nombreux travaux ont montré que la pseudo-aléatoire et les corrélations ne sont pas nécessairement un problème fondamental pour le calcul stochastique, si elles sont correctement prises en compte dans la conception du système [89]. Dans le cas particulier de notre machine bayésienne, nous avons constaté que nous pouvions nous appuyer sur des nombres aléatoires de faible qualité, générés par LFSR à faible coût, et obtenir une inférence bayésienne précise, à condition d'initialiser les LFSR avec des seeds bien choisies.

Notre conception exploite un LFSR de huit bits par colonne, qui génère des nombres pseudo-aléatoires avec une périodicité de seulement 255 cycles d'horloge. La Figure 2.4 met en évidence la corrélation entre les nombres aléatoires générés par les LFSR, en traçant les 255 valeurs générées par chaque LFSR en fonction des 255 valeurs générées par chaque autre LFSR en même temps. Cette figure est tracée avec des seeds de LFSR initialement choisies au hasard. Des corrélations évidentes sont observées entre certains LFSR. Ces corrélations font que certaines entrées des portes ET du calcul stochastique de la machine bayésienne sont corrélées, ce qui conduit aux écarts entre la loi de Bayes et les mesures de la Figure 3.8a.

Ce type de corrélation est observé avec la plupart des choix de seeds de LFSR. Cependant, le choix des seeds utilisées pour générer la Figure 2.5, qualifiées d'"optimales" par la suite, montre des corrélations considérablement réduites. Cela permet d'obtenir les résultats très précis de la Figure 3.8b, et suggère que ces seeds devraient être utilisées pour tous les calculs. En raison de la périodicité du LFSR, l'initialisation du LFSR doit être effectuée une seule fois, lorsque le système est mis en marche.

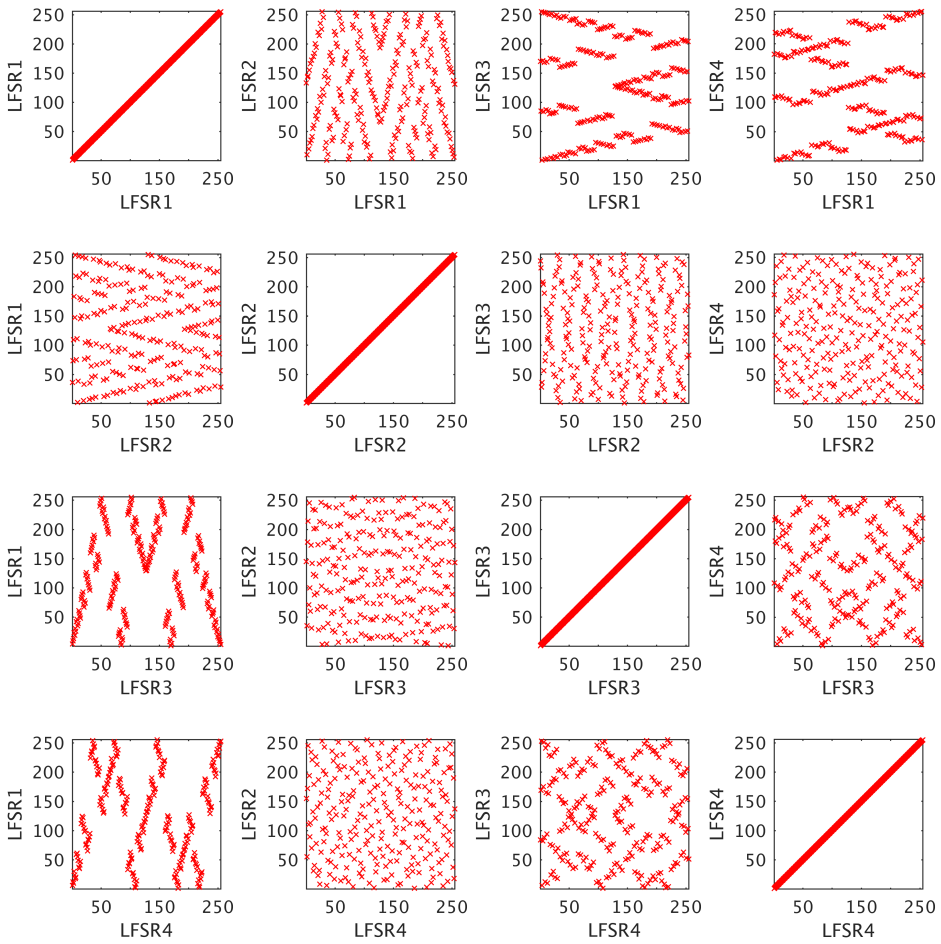


Figure 2.4: **Corrélations des nombres aléatoires générés par quatre LFSR, avec des graines initialement choisies au hasard** Chaque graphique présente la sortie de l'un des quatre LFSR de la machine bayésienne, en fonction de la sortie d'un autre LFSR. Chaque graphique contient 255 points correspondant aux 255 cycles de fonctionnement de la machine bayésienne. Les graphiques sur la diagonale (LFSR1/LFSR1, LFSR2/LFSR2, LFSR3/LFSR3, LFSR4/LFSR4) apparaissent comme des lignes x=y, par définition. La présence de motifs très discernables dans certains des graphiques (LFSR1/LFSR3, LFSR1/LFSR4) indique l'existence d'une forte corrélation entre la sortie de certains LFSR. D'un autre côté, les sorties de certains LFSR semblent largement non corrélées (LFSR1/LFSR2, LFSR2/LFSR3, LFSR2/LFSR4). Les graines pour les quatre LFSR sont, respectivement, en représentation hexadécimale : 50, E9, 10, et C6.

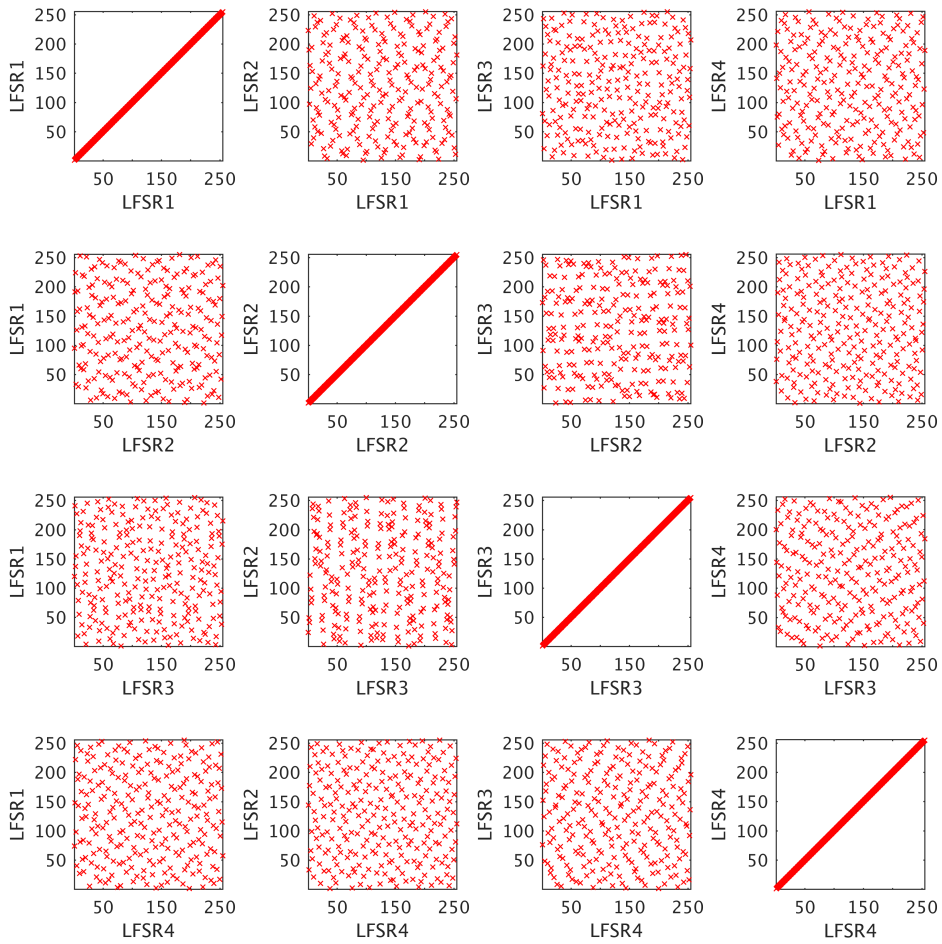


Figure 2.5: **Corrélations des nombres aléatoires générés par quatre LFSR, avec les graines optimales.** Cette Figure, tracée avec les mêmes conventions que la Fig.2.4, montre l'absence de corrélation évidente entre les sorties des différents LFSR. Les graines pour les quatre LFSR sont, respectivement, en représentation hexadécimale : EB, FB, 7F, et 5C.

2.3 Application Pratique : Reconnaissance des Gestes

Dans cette section, nous allons étudier une application pratique de la machine bayésienne stochastique avec la reconnaissance de gestes. Le principe de cette application est, comme je vais l'expliquer par la suite, de reconnaître un geste parmi quatre fait par une personne. Un ensemble de dix participants ont réalisé cette expérience et les données récoltées ont permis de réaliser plusieurs études comme le taux de reconnaissance des gestes, la résilience aux erreurs, l'explicabilité (i.e. montrer un geste inconnu et observer ce que la machine répond) ou encore l'utilisation de faibles données d'entraînement.

2.3.1 Fonctionnement de l'Application

2.3.1.1 Description de l'application

La tâche de reconnaissance de gestes est réalisée sur un ensemble de données collectées en laboratoire, comprenant dix participants. Chaque participant a été invité à effectuer quatre gestes (écrire les chiffres un, deux, trois et une signature spécifique à chaque personne) dans l'air. Aucune instruction spécifique n'a été donnée aux participants sur la manière d'effectuer les gestes, ce qui a entraîné une grande diversité au sein de l'ensemble de données. Les gestes ont été enregistrés à l'aide de l'accéléromètre triaxial d'une unité de mesure inertielle standard. Chaque participant a répété le même mouvement entre 25 et 27 fois. Le temps d'enregistrement variait selon le participant et le geste, allant de 1,3 à 3 secondes.

Dix caractéristiques, nommées F_0 à F_9, ont été extraites de chaque enregistrement, après avoir filtré la gravité en soustrayant les valeurs moyennes de l'enregistrement (voir les équations 2.6, 2.7, 2.9, 2.11) : accélération moyenne, accélération maximale sur les trois axes, variance de l'accélération sur les trois axes, valeur moyenne du jerk de l'accélération et valeur maximale du jerk de l'accélération sur les trois axes.

Nous avons entraîné le système en utilisant 20 des 25-27 enregistrements pour chaque participant, et les 5-7 derniers enregistrements ont été utilisés pour le tester. Notre modèle suppose les priors uniformes, ainsi l'équation 2.2 se simplifie en enlevant le terme $p(A)$ ce qui nous permet d'enlever un bloc mémoire comprenant ces priors. L'entraînement consiste à ajuster les vraisemblances $p(O_n|Y = y)$ aux données d'entraînement. Étant donné que les données d'entraînement sont très limitées, nous avons ajusté ces vraisemblances par des fonctions gaussiennes (en utilisant la fonction fitdist de MathWorks MATLAB). Nous avons ensuite discrétisé les distributions gaussiennes résultantes aux 512 valeurs d'entrée possibles des observations. Ensuite, nous avons normalisé les probabilités dans chaque colonne de la machine bayésienne afin que la valeur maximale soit égale à un. Enfin, nous avons quantifié les valeurs de probabilité normalisées en entiers de huit bits, avec la valeur zéro équivalente à 1/256 et 255 à 256/256.

Pour optimiser davantage la consommation d'énergie du système, nous utilisons unique-

ment six des dix caractéristiques extraites dans la machine bayésienne. Suite à une étude systématique, les caractéristiques supprimées pour l'expérience sont F_0, F_2, F_3, F_8. De plus, nous avons constaté qu'élargir les gaussiennes obtenues lors de l'ajustement des données permettait à le calcul stochastique de converger plus rapidement, permettant au système d'atteindre une meilleure précision. Par conséquent, dans tous nos résultats, l'écart type de la gaussienne dans les vraisemblances ajustées est multiplié par un coefficient d'élargissement de 1,3 par rapport à l'ajustement initial.

Nous avons également développé une approche soucieuse de la consommation énergétique que nous appelons "power-conscious" qui au lieu de compter l'ensemble des 'uns' sortis par la machine bayésienne, opération faite dans l'approche conventionnelle, va arrêter le calcul après que le premier 'un' soit sorti. Dans la technique conventionnelle, la réponse de la machine bayésienne à une sortie est donnée par l'argmax du nombre de 'uns' produits par chaque ligne. Dans la méthode power-conscious, la réponse est la ligne qui a produit le premier 'un'. Dans les deux cas, si aucune sortie n'a produit un 'un' la réponse est considérée comme une erreur.

2.3.1.2 Calcul des features

La première caractéristique F_0 est la racine carrée de la valeur moyenne de l'accélération au carré

$$F_0 = \sqrt{\frac{1}{T}\sum_t \left(\ddot{x}^2(t) + \ddot{y}^2(t) + \ddot{z}^2(t)\right)}. \tag{2.6}$$

Les caractéristiques F_1, F_2 et F_3 représentent l'accélération maximale le long de chaque axe :

$$\begin{aligned} F_1 &= \max_t |\ddot{x}(t)|/F_0 \\ F_2 &= \max_t |\ddot{y}(t)|/F_0 \\ F_3 &= \max_t |\ddot{z}(t)|/F_0. \end{aligned} \tag{2.7}$$

F_4 et F_5 sont calculés en utilisant la variance de l'accélération sur les trois axes :

$$\begin{aligned} \mathrm{Var}\,\ddot{x} &= \frac{1}{T}\sum_t \ddot{x}^2(t) \\ \mathrm{Var}\,\ddot{y} &= \frac{1}{T}\sum_t \ddot{y}^2(t) \\ \mathrm{Var}\,\ddot{z} &= \frac{1}{T}\sum_t \ddot{z}^2(t). \end{aligned} \tag{2.8}$$

Nous classons ensuite ces trois variances comme "min" pour la plus petite, "inter" pour la

moyenne et "max" pour la plus grande, et calculons F_4 et F_5 comme suit :

$$\begin{aligned} F_4 &= \sqrt{\min \mathrm{Var}\,\ddot{x}, T\,\mathrm{Var}\,\ddot{y}, \mathrm{Var}\,\ddot{z} / \max \mathrm{Var}\,\ddot{x},\ \mathrm{Var}\,\ddot{y}, \mathrm{Var}\,\ddot{z}} \\ F_5 &= \sqrt{\mathrm{inter}\,\mathrm{Var}\,\ddot{x}, \mathrm{Var}\,\ddot{y}, \mathrm{Var}\,\ddot{z} / \max \mathrm{Var}\,\ddot{x},\ \mathrm{Var}\,\ddot{y}, \mathrm{Var}\,\ddot{z}}. \end{aligned} \tag{2.9}$$

Les dernières caractéristiques sont basées sur les séries dérivées de l'accélération, appelées "jerk", calculées par différence finie et avec un filtrage (nous prenons comme convention que les valeurs avant et après l'enregistrement sont égales à zéro) :

$$\begin{aligned} \dddot{x}(t) &= \frac{\ddot{x}(t+1) - \ddot{x}(t-1)}{2} \\ \dddot{y}(t) &= \frac{\ddot{y}(t+1) - \ddot{y}(t-1)}{2} \\ \dddot{z}(t) &= \frac{\ddot{z}(t+1) - \ddot{z}(t-1)}{2} \\ jerk_k(t) &= 0.4 \times \dddot{k}(t) + 0.3 \times (\dddot{k}(t-1) + \dddot{k}(t+1)) \quad \text{avec k = x; y; z} \end{aligned} \tag{2.10}$$

Les caractéristiques F_6 à F_9 sont ensuite calculées de manière similaire à F_0 à F_3, avec le jerk remplaçant l'accélération :

$$\begin{aligned} F_6 &= \sqrt{\frac{1}{T}\sum_t \left(jerk_x^2(t) + jerk_y^2(t) + jerk_z^2(t)\right)} / F_0 \\ F_7 &= \max t |jerkx(t)| \frac{F_0}{F_6} \\ F_8 &= \max t |jerky(t)| \frac{F_0}{F_6} \\ F_9 &= \max t |jerkz(t)| \frac{F_0}{F_6}. \end{aligned} \tag{2.11}$$

2.3.2 Analyse des Résultats

2.3.2.1 Résultats sans Erreurs

La Figure 2.6 montre les résultats de simulations de la machine bayésienne sur la tâche de reconnaissance de gestes en utilisant les deux méthodes de calcul stochastique : la méthode conventionnelle et la méthode power-conscious. Pour ces deux méthodes, les résultats sont montrés pour un nombre variable de cycles d'horloge afin d'évaluer les performances de reconnaissance à nombre de cycles plus faibles et énergétiques qui seront meilleures plus le nombre de cycles effectués est faible. L'analyse en détail se fera dans le chapitre 3.

Nous pouvons remarquer que dans le graphe de la figure 2.6, la précision augmente rapidement pour ensuite rester stable à partir de 50 cycles pour les deux méthodes. La méthode conventionnelle est plus précise mais cela est normal car nous faisons la somme de tous les 'uns' qui sortent. Cependant en terme d'écart de précision, seulement 3% séparent les deux

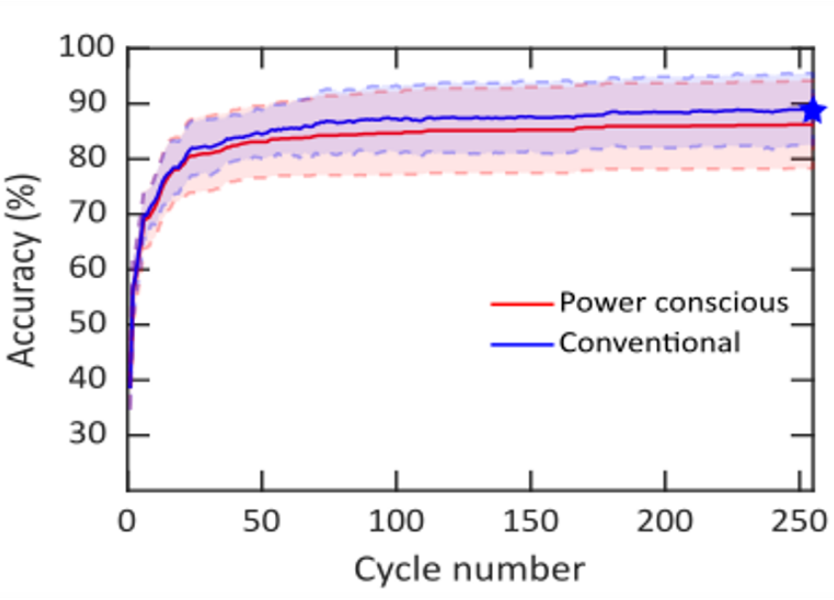


Figure 2.6: **Résultats de la machine bayésienne à une tâche pratique de reconnaissance de gestes.** *Précision moyenne* en fonction du nombre de cycles pour deux types de calcul : en utilisant une méthode économe en énergie en ne prenant en compte que le premier '1' pour la décision (en rouge) et en utilisant le calcul stochastique conventionnel en utilisant le nombre maximum de '1' pour la décision (en bleu). Les ombres autour du graphique montrent un écart-type de la précision moyenne sur les dix sujets.

méthodes (89% pour la méthode conventionnelle et 86% pour la méthode power-conscious) ce qui est surprenant mais qui peut s'expliquer simplement par l'apparition du premier 'un' qui signifie une 'tendance' dans cette valeur de sortie et qui va donc continuer si plus de cycles sont effectués à donner des 'uns'.

2.3.2.2 Résultats avec Erreur Mémoire

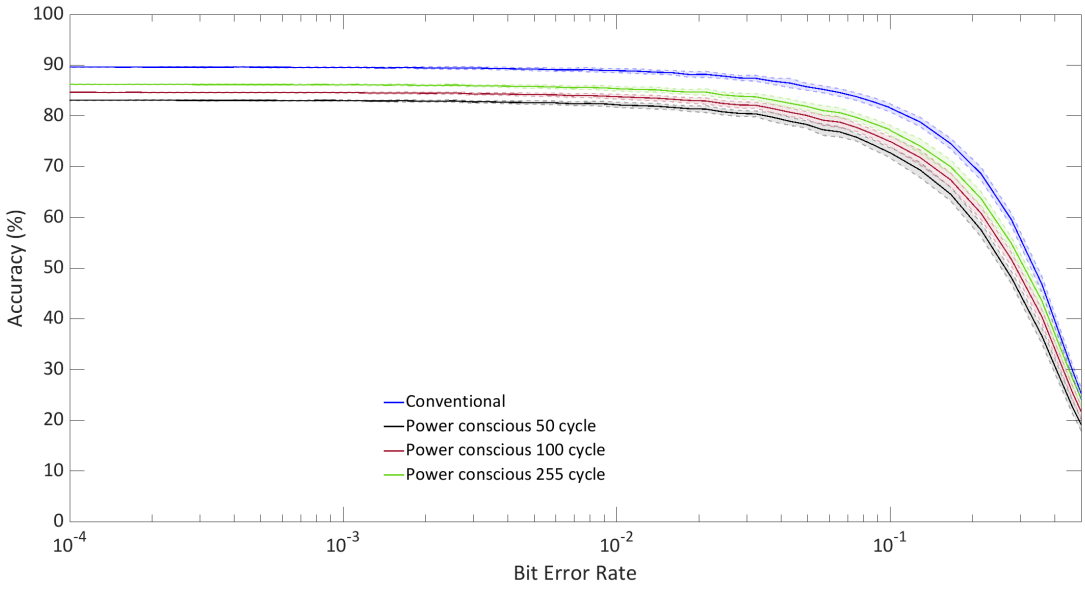


Figure 2.7: **Impact des erreurs sur les bits de mémoire sur la précision de la tâche de reconnaissance des gestes**. Précision moyenne simulée sur la tâche de reconnaissance des gestes, lorsque des erreurs de bits ont été artificiellement introduites dans les bits de mémoire, en fonction du taux d'erreur de bit de mémoire. La simulation a été répétée 40 fois, les ombres autour du graphique montrent l'écart type de la précision moyenne le long de ces répétitions. Les résultats sont présentés pour le mode power-conscious avec un seuil à 50, 100, et 255 cycles, et pour le mode de calcul stochastique conventionnel utilisant tous les 255 cycles.

Cependant, pour des raisons fondamentales, il est important de comprendre l'impact que les erreurs de bits dans la mémoire auraient sur le fonctionnement de la machine. La Figure 2.7 montre les résultats de simulations de la machine bayésienne à grande échelle sur la tâche de reconnaissance de gestes, où des erreurs de bits ont été artificiellement ajoutées dans les tableaux de mémoire de vraisemblance. Les résultats sont rapportés à la fois en utilisant la stratégie conventionnelle de calcul stochastique avec 255 cycles, et la stratégie power-conscious, qui arrête le calcul dès qu'une ligne fournit une sortie à un. Les résultats de la technique power-conscious sont montrés pour des calculs avec un nombre maximum variable de cycles d'horloge. La précision sur la reconnaissance des gestes montre une grande résilience à ces erreurs de bits, à la fois dans les cas du calcul stochastique conventionnel et power-conscious. Même pour des taux d'erreur de bits aussi élevés que 10^{-2}, la précision est réduite de seulement 0,7 points de pourcentage avec la stratégie de calcul stochastique conventionnel. Ce résultat est similaire à ce qui a été observé dans les réseaux neuronaux matériels [8] et semble surprenant au premier abord : les erreurs de bits peuvent affecter les bits les plus significatifs des valeurs stockées, et donc changer les vraisemblances stockées de manière très

significative. La tolérance peut être liée à deux raisons :

- Premièrement, la nature stochastique du calcul signifie qu'une majorité de lignes donnent une valeur zéro. Les erreurs de mémoire dans de telles lignes n'ont généralement aucun impact sur le résultat final.
- Deuxièmement, les différentes caractéristiques utilisées comme entrées de la machine bayésienne portent des informations partiellement redondantes. Par conséquent, une erreur dans une vraisemblance peut être compensée par les vraisemblances concernant les autres entrées, fournissant une grande résilience à la machine bayésienne.

2.3.2.3 Résultats avec Erreur dans le Flux de Bits

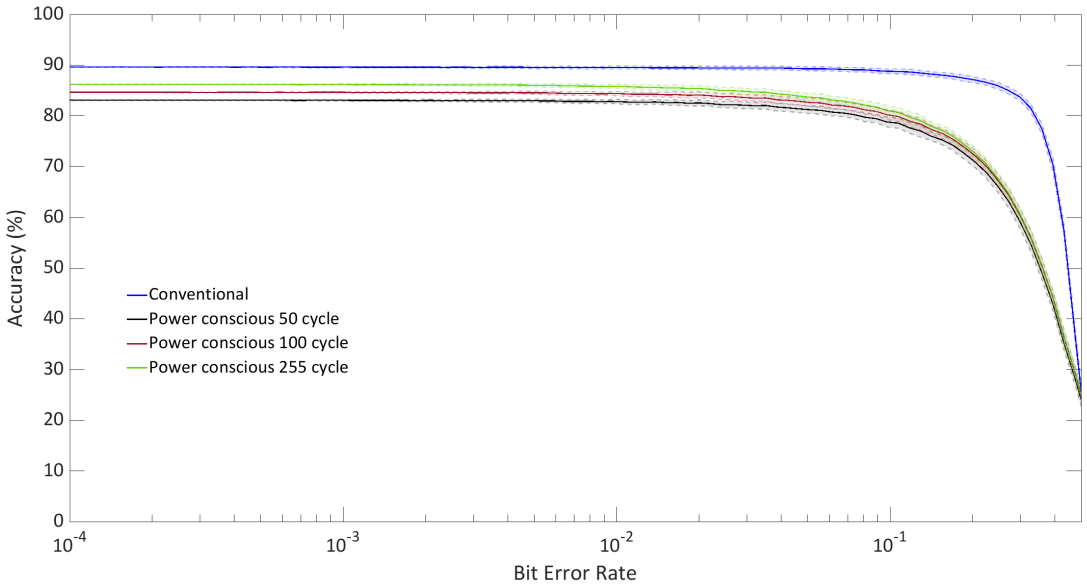


Figure 2.8: **Impact des erreurs uniques sur la précision de la tâche de reconnaissance des gestes**. Précision moyenne simulée sur la tâche de reconnaissance des gestes, lorsque des erreurs uniques ont été artificiellement introduites dans le calcul, en fonction du taux d'erreur de bit de mémoire. Les erreurs uniques sont introduites à la sortie des circuits Gupta (c'est-à-dire sur les flux de bits stochastiques créés par chaque bloc de vraisemblance). La simulation a été répétée 100 fois, les ombres autour du graphique montrent un écart type de la précision moyenne le long de ces répétitions. Les résultats sont présentés pour le mode power-conscious avec un seuil à 50, 100, et 255 cycles, et pour le mode de calcul stochastique conventionnel utilisant tous les 255 cycles.

La machine bayésienne fonctionne sur les principes du calcul stochastique, ce qui signifie que la décision finale est basée sur les statistiques sur plusieurs cycles d'horloge. Ce mode de fonctionnement signifie que les erreurs de bits n'ont généralement pas d'effets catastrophiques et sont naturellement moyennées. Cette caractéristique confère à la machine bayésienne une immunité naturelle aux erreurs uniques (Single Event Upset : SEU), qui peuvent se produire dans un environnement riche en radiations, en particulier lors de l'exploitation de la machine à basse tension d'alimentation.

La Figure 2.8 montre des simulations de la tâche de reconnaissance des gestes où des erreurs uniques ont été artificiellement introduites à la sortie des blocs de vraisemblance. Les simulations sont présentées en utilisant l'opération du calcul stochastique conventionnel avec 255 cycles et l'approche power-conscious avec plusieurs nombres maximum de cycles d'horloge. La précision montre une immunité exceptionnelle aux erreurs uniques, en particulier dans l'opération de calcul stochastique conventionnel. Même avec 10% d'erreurs de bits, la précision finale est à peine affectée. L'immunité est largement améliorée par rapport à l'impact

des erreurs de bits de mémoire (Fig. 2.7), car, contrairement au cas de la mémoire, une erreur de bit a un impact sur un seul cycle et peut être moyennée par les cycles suivants. Le mode power-conscious, qui arrête le calcul dès qu'un un est produit à n'importe quelle sortie, est très résistant aux erreurs uniques, mais moins que le mode conventionnel. Ce résultat est naturel, car les erreurs uniques peuvent alors produire un un incorrect et arrêter le calcul sur une mauvaise réponse. Dans des contextes avec des niveaux élevés d'erreurs uniques (environnements extrêmes), le calcul stochastique conventionnel devrait donc être préféré.

2.3.3 Vers une Explicabilité Accrue

Un avantage très significatif des approches bayésiennes est leur explicabilité. Dans le cas de notre machine bayésienne, il suffit de regarder l'écart type de la vraisemblance associée à chaque entrée pour comprendre quelles caractéristiques sont importantes pour la décision (faible écart type) et lesquelles sont accessoires (écart type élevé).

L'explicabilité est souhaitable dans de nombreuses situations critiques pour des raisons éthiques et réglementaires [91]. De plus, le fait que l'inférence bayésienne prenne des décisions basées sur des modèles explicables a des conséquences très pratiques, en particulier lorsque nous les utilisons avec des entrées qui diffèrent de celles utilisées pour entraîner le modèle. Dans ce type de situation, les réseaux neuronaux, qui fonctionnent en ajustant les données d'entraînement, ont tendance à donner des réponses sûres et imprévisibles. En raison de ce problème, les réseaux neuronaux donnent facilement des réponses clairement fausses avec une apparence de haute confiance lorsqu'ils sont présentés avec des données hors distribution [92, 93]. En revanche, les modèles bayésiens excellent à reconnaître les situations où ils ne sont pas en mesure de fournir une réponse fiable [94].

Pour illustrer cette capacité dans le cas de notre machine bayésienne, nous présentons des résultats sur le comportement des machines bayésiennes entraînées à la reconnaissance des gestes, lorsqu'elles sont présentées avec des données d'un autre sujet que celui qu'elles ont été formées à reconnaître. Nous montrons que dans une grande majorité de ces cas, la machine identifie correctement qu'elle ne peut pas fournir une réponse fiable. Plus précisément, il peut y avoir deux signatures que la machine bayésienne ne peut pas fournir une réponse confiante :

- Il semble que, lorsque certaines observations de l'entrée sont considérées comme particulièrement improbables de se produire simultanément dans les données d'entraînement, il est possible que toutes les rangées de la machine bayésienne ne produisent que, ou presque que, des 'zéros'. Il se pourrait que cette situation survienne, par exemple, en cas de défaillance d'un capteur, où la sortie pourrait devenir incohérente avec les autres lectures de capteurs. On peut supposer que ce type d'incertitude soit l'incertitude épistémologique.

- Il est possible que, si les observations ne sont pas la signature d'une situation particulière, il pourrait y avoir une chance que toutes les rangées produisent un nombre équilibré de 'uns'. On pourrait envisager que cette situation corresponde à une distribution de probabilité uniforme, c'est-à-dire à une situation où, théoriquement, l'entropie serait maximale, et il semble que le modèle n'exprimerait pas de préférence forte pour une réponse. On pourrait supposer que c'est là un exemple d'incertitude aléatorique.

Sur la base de ces considérations, nous pouvons définir une métrique pour déterminer si la machine bayésienne est confiante quant à sa sortie, la Figure 2.9a illustre sur des exemples comment ce traitement est effectué :

- Concernant le premier critère, nous comptons le nombre de 'uns' pour les quatre rangées. Si le nombre de 'uns' sur toutes les rangées est inférieur à une certaine proportion T du nombre maximal de cycles (ici 255 cycles), que nous appelons seuil de certitude, nous considérons que la machine bayésienne est incertaine.

- Concernant le deuxième critère, nous calculons la somme des 'uns' pour chaque rangée et nous divisons la somme sur la rangée avec le plus grand nombre de 'uns' par la somme totale sur les quatre rangées. La machine est considérée comme fournissant un résultat certain si ce ratio est supérieur à 50%.

La Figure 2.9b montre graphiquement la puissance de cette évaluation de l'incertitude. Cette Figure résume les résultats de plusieurs expériences. Nous avons présenté chaque geste de notre ensemble de données aux machines bayésiennes formées pour chaque sujet. L'abscisse est séparée en dix sections correspondant aux dix sujets, et pour chaque sujet, les quatre points sont respectivement la signature, les chiffres un, deux et trois. L'ordonnée correspond aux dix machines bayésiennes formées pour chaque sujet, et les quatre points de données représentent ses quatre sorties (la signature et les chiffres un, deux, trois). Chaque point dans la carte de couleurs indique la proportion de gestes correspondant à l'abscisse qui a été reconnue avec confiance comme la sortie de l'ordonnée. Pour cette Figure, un seuil de certitude de 10% a été utilisé.

Le résultat souhaitable est que lorsque les sujets d'entrée et de sortie ne correspondent pas, la carte de couleurs apparaisse toute blanche (ce qui signifie que la machine n'a fourni aucun résultat certain). C'est effectivement ce qui est largement observé. Une exception est la machine bayésienne pour le sujet six, qui a tendance à reconnaître un geste pour les sujets sept, huit et neuf. Inversement, lorsque le sujet d'entrée et la machine bayésienne correspondent, le résultat souhaité est une carte de couleurs toute noire sur les diagonales et blanche ailleurs. Les résultats sont très proches de ce comportement idéal, avec quelques points gris indiquant les erreurs de la machine bayésienne.

Le Tableau 2.1 détaille ces résultats quantitativement. Il illustre également comment le comportement de la machine bayésienne peut être modifié en ajustant le seuil de certitude. Lorsqu'un geste correspondant au mauvais sujet est présenté, la machine bayésienne déclare correctement l'incertitude 94% à 98% du temps, en fonction du seuil de certitude. Lorsque des gestes du bon sujet sont présentés, la proportion de temps où la machine fait une véritable erreur (c'est-à-dire qu'elle identifie le mauvais geste tout en étant certaine) est de 5% pour un seuil de certitude de zéro, et peut être réduite à 3% avec un seuil de certitude de 10%. Cependant, cela se fait au prix de plus de gestes signalés comme incertains. Ces résultats illustrent comment un opérateur peut ajuster le comportement de la machine bayésienne, en privilégiant l'élimination des réponses incorrectes, ou la minimisation des réponses incertaines. Ce degré de liberté peut être particulièrement utile pour les tâches médicales où une intervention injustifiée due à une mauvaise réponse peut avoir plus de coût qu'une non-intervention basée sur une réponse incertaine.

Cas où un geste est présenté à la machine bayésienne **correspondant à un sujet différent**

x	T=0	T=5%	T=10%
La machine bayésienne était incertaine (comportement souhaité)	93%	97%	98%
La machine bayésienne a fourni une sortie certaine	7%	3%	2%

Cas où un geste est présenté à la machine bayésienne **correspondant au sujet approprié**

	T=0	T=5%	T=10%
La machine bayésienne était certaine du bon geste (comportement souhaité)	88%	80%	71%
La machine bayésienne était certaine d'un geste incorrect	7%	6%	5%
La machine bayésienne était incertaine	5%	14%	24%

Table 2.1: **Comportement quantitatif de la machine bayésienne lorsque les gestes d'entrée et les gestes reconnus peuvent ne pas correspondre au sujet.** Les résultats sont présentés pour trois valeurs du seuil de certitude T.

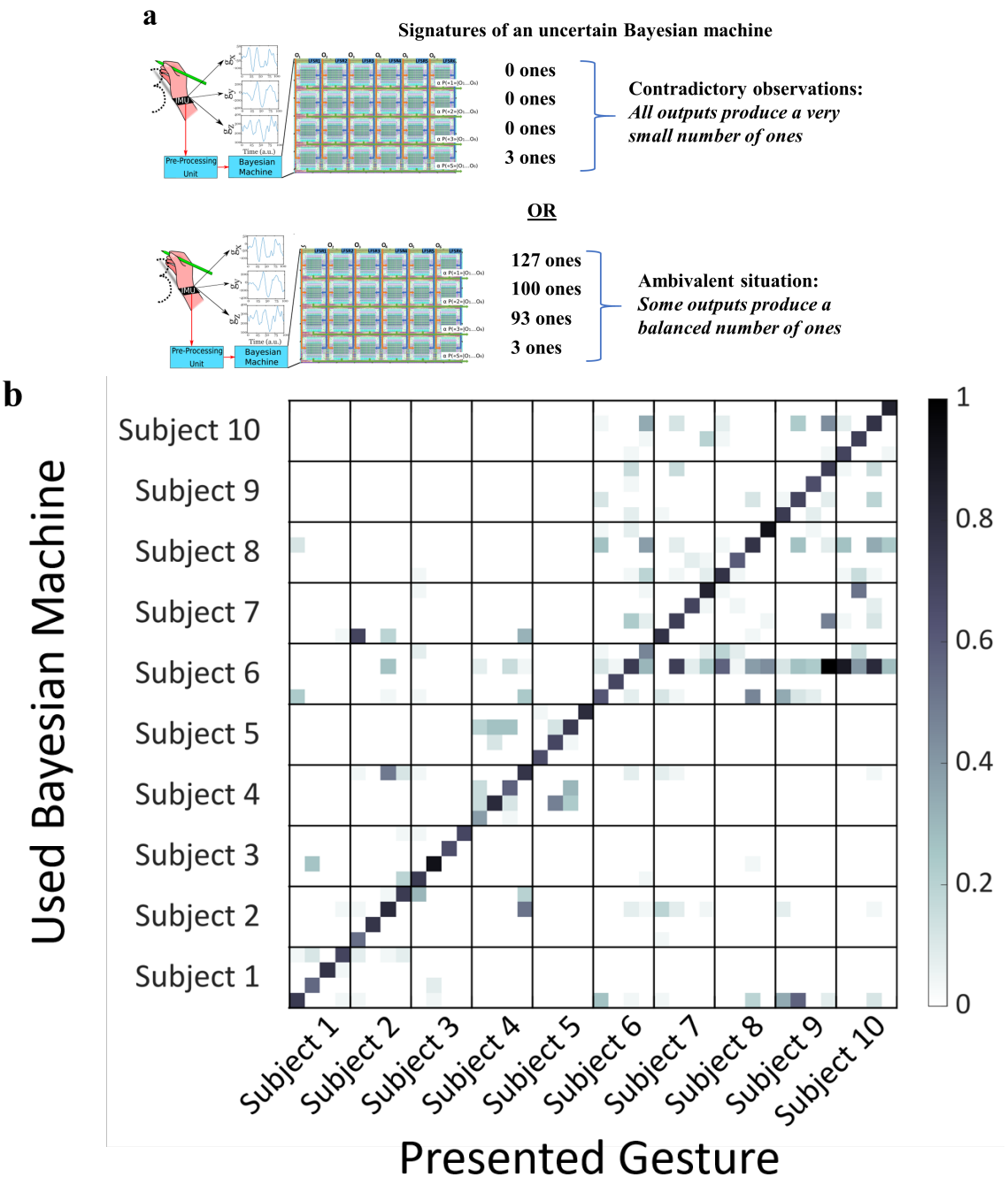


Figure 2.9: **Capacité de la machine bayésienne à identifier les situations incertaines.** **a** Illustration schématique des deux signatures d'une machine bayésienne incertaine. **b** Réponse moyenne simulée de la machine bayésienne, pour la tâche de reconnaissance de gestes, dans des situations où un geste présenté à la machine bayésienne peut provenir d'un sujet différent de celui sur lequel la machine a été entraînée. Abscisse : geste présenté. Pour chaque sujet, les quatre points représentent les gestes un, deux, trois, et la signature. Ordonnée : machine bayésienne utilisée. Les différents points représentent les différentes sorties de la machine dans le même ordre que l'abscisse. Couleur : proportion de cas conduisant à la sortie dans l'ordonnée pour les gestes présentés correspondant à l'abscisse, avec la machine bayésienne fournissant une sortie certaine. Les résultats sont présentés pour un seuil de certitude T de 10%.

2.3.4 Performances sur des Ensembles de Données Réduits

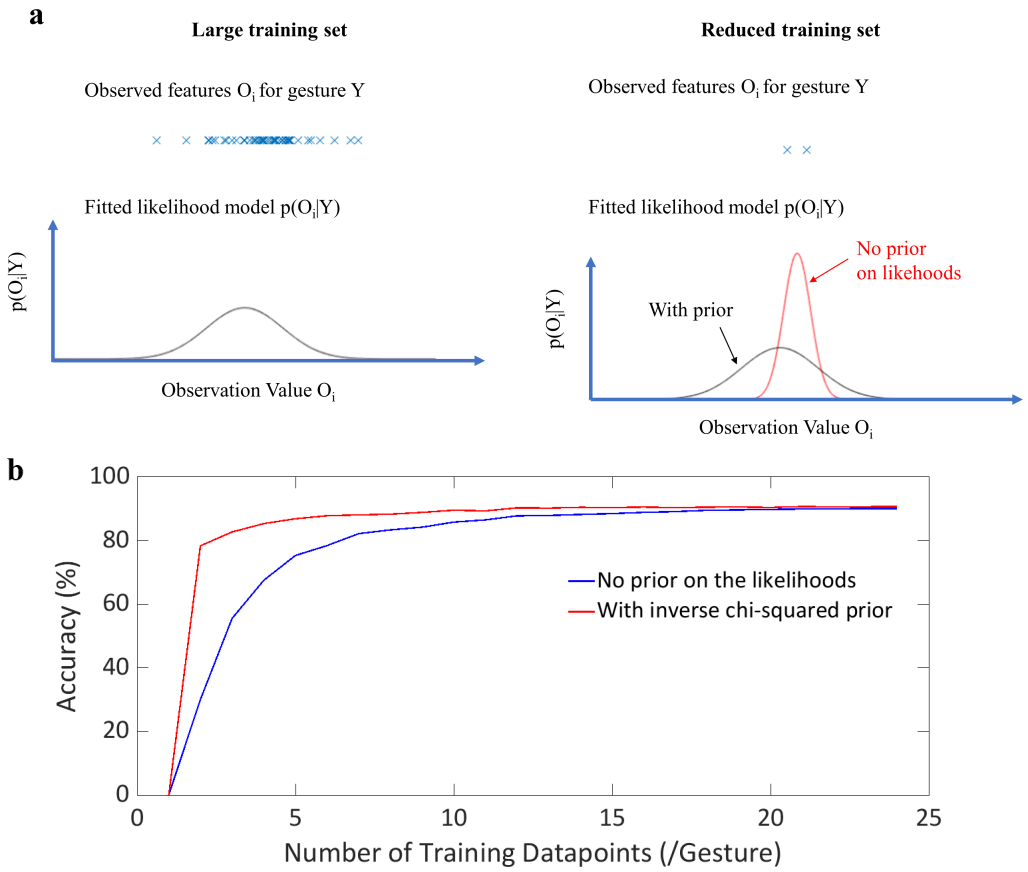


Figure 2.10: **Impact de la taille de l'ensemble de données d'entraînement sur la précision de la machine bayésienne.** **a** Illustration du défi des petites données. Lors de l'utilisation d'un grand ensemble de données d'entraînement représentatif, les modèles de vraisemblance peuvent être ajustés directement aux données d'entraînement. En revanche, les ensembles de données réduits peuvent ne pas être représentatifs, et l'ajustement direct d'un modèle de vraisemblance aux données peut conduire à des résultats dénués de sens. L'incorporation d'un a priori lors de l'ajustement de la vraisemblance peut cependant résoudre ce problème. Cette possibilité est un avantage important des approches bayésiennes. **b** Précision moyenne simulée des machines bayésiennes en fonction du nombre d'exemples utilisés dans l'ensemble de données d'entraînement. La précision est tracée dans une situation où les modèles de vraisemblance sont des ajustements gaussiens des données d'entraînement, et lorsqu'un a priori normal inverse chi-carré a été utilisé. Tous les résultats sont présentés en utilisant la méthode conventionnelle de calcul stochastique avec 255 cycles. Tous les résultats sont également validés par croisement, avec 20 séparations différentes entre les ensembles d'entraînement et de test.

Un avantage important des approches bayésiennes est qu'elles s'adaptent bien à la quantité de données d'entraînement. L'ajustement des modèles d'apprentissage machine lorsque très peu de données d'entraînement sont disponibles est un défi majeur, car la base d'entraînement n'est pas représentative. Dans cette situation, les modèles d'apprentissage machine ont tendance à "sur-apprendre" les quelques exemples d'entraînement qui sont donnés, et à donner des prédictions dénuées de sens lorsque des données qui diffèrent des exemples d'entraînement sont fournies. Heureusement, les modèles bayésiens peuvent gérer particulièrement bien ce type de situation, grâce à leur explicabilité, qui permet d'incorporer des hypothèses a priori dans le processus d'ajustement.

Par exemple, la Figure 2.10a illustre le défi de l'entraînement de la machine bayésienne sur la tâche de reconnaissance de gestes dans une situation de très petites données. Lorsque suffisamment de données d'entraînement sont disponibles, les modèles de vraisemblance gaussiens peuvent être directement ajustés à ces données. Lorsque très peu de données sont disponibles, l'ajustement d'une gaussienne aux données donne facilement un résultat dénué de sens. Une solution est d'incorporer des hypothèses a priori sur ce que l'on attend comme valeur moyenne et écart type. Dans le domaine des statistiques bayésiennes, une technique établie pour faire cela est d'inférer les paramètres du modèle de vraisemblance à partir des données d'entraînement en utilisant un a priori sur le modèle. Si nous voulons ajuster un modèle gaussien avec une moyenne μ et une variance σ^2 à N points de données x_i, l'inférence bayésienne suggère que la distribution de probabilité sur les paramètres du modèle étant donné les valeurs du point de données est

$$p(\mu,\sigma^2|x_1,...,x_N) \propto p(x_1,...,x_N|\mu,\sigma^2) \times p(\mu,\sigma^2) \propto \prod_{i=1}^{N} exp(-(x_i-\mu)^2/2\sigma^2) \times p(\mu,\sigma^2), \quad (2.12)$$

où $p(\mu,\sigma^2)$ exprime la croyance a priori sur les valeurs de μ et σ^2. Un choix relativement établi[95] pour cet a priori est la distribution normale inverse chi-carrée (ou NIX) :

$$p(\mu,\sigma^2) \propto \left(\frac{1}{\sigma^2}\right)^{(\nu_0+3)/2} exp\left(-\frac{\nu_0\sigma_0^2+\kappa_0\left(\mu-m_0\right)^2}{2\sigma^2}\right). \quad (2.13)$$

Dans cet a priori, m_0 exprime la valeur moyenne estimée et κ_0 le degré de confiance à son sujet, σ_0^2 la valeur estimée de la variance, et ν_0 le degré de confiance à son sujet. La distribution normale inverse chi-carrée est l'a priori conjugué pour la distribution normale qui permet de dériver des expressions en forme fermée : en combinant les équations(2.12) et (2.13), on peut montrer (voir[95], p.136) que l'espérance pour μ est

$$\bar{\mu} = \frac{\kappa_0 m_0 + \sum_{i=1}^{N} x_i}{\kappa_0 + N}. \quad (2.14)$$

L'espérance pour σ^2 est

$$\bar{\sigma^2} = \frac{1}{\kappa_0 + N}\left(\nu_0\sigma_0^2 + \sum_{i=1}^{N}(x_i - \bar{x})^2 + \frac{N\kappa_0}{\kappa_0 + N}(m_0 - \bar{x})^2\right), \tag{2.15}$$

où $\bar{x} = (\sum_{i=1}^{N} x_i)/N$.

Les équations(2.14) et (2.15) peuvent être utilisées pour entraîner des modèles de vraisemblance dans une situation avec un faible nombre de données. Pour notre travail, nous normalisons chaque caractéristique, avec zéro étant la valeur la plus basse dans l'ensemble de données d'entraînement de petites données et un la valeur la plus élevée. Nous choisissons comme a priori $m_0 = 0.5$, $\sigma_0^2 = 0.25$, avec une confiance $\kappa_0 = 1$, $\nu_0 = 1$.

La Figure 2.10b montre la précision moyenne pour la tâche de reconnaissance de gestes, en fonction du nombre d'exemples de gestes qui ont été utilisés pour l'entraînement. Les résultats sont validés par croisement, ce qui signifie que 20 séparations différentes entre les ensembles d'entraînement et de test sont effectuées pour chaque situation. Les résultats sont présentés lorsque les vraisemblances sont simplement ajustées aux données, et lorsque cette opération est effectuée avec un a priori en utilisant les équations(2.14) et (2.15). Lors de l'utilisation d'un a priori, une précision de 78% est obtenue en utilisant seulement deux échantillons par geste, une précision de 87% est obtenue en utilisant cinq échantillons. Sans l'a priori, respectivement sept et douze échantillons par geste sont nécessaires pour obtenir les mêmes précisions. Lorsque le nombre d'exemples d'entraînement est augmenté, les avantages de l'a priori disparaissent progressivement : pour plus de 20 exemples, l'utilisation de l'a priori n'apporte aucun avantage discernable. Ces résultats mettent en évidence l'excellente capacité à gérer les situations de petites données, ce qui peut être particulièrement utile, par exemple, pour les applications médicales.

Conclusion Dans ce chapitre, nous avons exploré en détail le fonctionnement des machines bayésiennes, mettant en lumière les principes fondamentaux et les techniques essentielles de l'approche bayésienne. Nous avons également montré comment réaliser une machine bayésienne en utilisant le calcul stochastique pour effectuer les multiplications des probabilités et souligné l'importance du choix des seeds. Cette utilisation du calcul stochastique nous a permis de montrer la résilience de la machine face aux erreurs dans la mémoire et aux erreurs dans le flux de bits.

Puis avec l'application de reconnaissance de gestes, nous avons pu montrer que la machine bayésienne donnait des prédictions explicables. L'explicabilité est particulièrement souhaitable dans de nombreuses situations critiques, tant pour des raisons éthiques que réglementaires. Le fait que l'inférence bayésienne prenne des décisions basées sur des modèles explicables a également des conséquences pratiques importantes, notamment dans les situations où les entrées diffèrent de celles utilisées pour l'entraînement du modèle. Par exemple, notre machine bayésienne excelle à reconnaître les situations où elle ne peut fournir une réponse fiable, une

caractéristique particulièrement utile pour les dispositifs médicaux où des décisions erronées peuvent avoir des conséquences dramatiques.

Enfin, nous avons également illustré la possibilité d'entraîner la machine bayésienne avec peu de données, tout en obtenant une précision moyenne de 78% sur la reconnaissance des gestes en utilisant seulement deux exemples par geste. Cette capacité à généraliser et à incorporer des hypothèses préalables sur le modèle dans le processus d'entraînement de la vraisemblance souligne le potentiel des machines bayésiennes dans divers domaines d'application, et pose les bases pour les discussions ultérieures sur la conception et le développement des puces bayésiennes dans le chapitre suivant.

Chapter 3

Machine Bayésienne II, le retour du roi

Intégration de la machine Bayésienne dans un circuit comprenant des memristors

> Nous y sommes enfin, la grande bataille de notre temps.
>
> Gandalf, Le Seigneur des anneaux : le retour du roi

DANS CE CHAPITRE, également adapté de l'article [85], je présente la puce stochastique fabriquée, notamment son fonctionnement, en programmation, lecture et inférence. Puis j'analyse les mesures réalisées avec cette puce dans deux versions : la version testée sous pointes et la version testée dans un boitier. Enfin, je réalise une simulation de l'estimation énergétique de la puce en me basant sur l'application de reconnaissance de gestes vue précédemment.

3.1 Design et Fabrication de la Puce Bayésienne

3.1.1 Processus de Fabrication

Afin de valider la faisabilité de l'inférence bayésienne basée sur les memristors, nous avons fabriqué un circuit prototype dans un processus hybride CMOS/RRAM. La partie CMOS du circuit est fabriquée à l'aide d'un procédé 130 nanomètres de fonderie à faible consommation d'énergie, comprenant quatre couches de métaux. Les memristors sont fabriqués au-dessus des vias exposés et sont composés d'un empilement TiN/HfO_x/Ti/TiN. L'HfO_x actif est déposé par dépôt en couches atomiques et a une épaisseur de 10 nanomètres. La couche de Ti a également une épaisseur de 10 nanomètres, et la structure du memristor a un diamètre de 300 nanomètres. Au-dessus des memristors, une cinquième couche de métal est déposée. Les pads d'entrée/sortie sont alignés en une rangée de 25 pads conçue pour la caractérisation par une carte à pointe personnalisée. Le noeud 130 nanomètres constitue un point économiquement avantageux en termes de niveaux de tension supportés avec diverses options de transistors MOS, ce qui est approprié pour les circuits analogiques et mixtes à faible consommation d'énergie avec des mémoires intégrées, ciblant les applications Internet des objets (IoT) et de l'intelligence artificielle sur puce (edge AI).

La figure 3.1a montre la puce fabriquée avec la structure superposée de la machine bayésienne. La puce de test met en œuvre un système avec 16 matrices de mémoire de vraisemblance, organisées en quatre lignes et quatre colonnes, connectées par des fils horizontaux et verticaux, et contrôlées par une unité de contrôle numérique intégrée. Elle a été conçue à l'aide d'un flux de conception fait maison permettant la distribution des blocs de memristors. La topologie de la puce suit étroitement le schéma conceptuel de la machine bayésienne de la figure 2.2b. Les figures 3.1b-c montrent les détails de l'une des matrices de mémoire de vraisemblance et son circuit périphérique (la figure 3.1e montre le schéma associé). La figure 3.1d montre une image de microscopie électronique d'un memristor intégré dans le back-end de la puce.

Notre processus hybride présente certaines contraintes en raison de sa nature partiellement académique : seules quatre couches de métaux sont disponibles pour l'interconnexion, la cinquième couche étant utilisée uniquement pour accéder aux memristors. De plus, le circuit est testé à l'aide de pointes. Cette puce de test nous permet néanmoins de démontrer tous les défis associés à la fabrication de la machine bayésienne.

3.1.2 Structure 2T2R

La variabilité des dispositifs constitue une forte limitation de la technologie des memristors, en particulier pour le calcul en mémoire et proche mémoire, où les codes de correction d'erreur puissants ne peuvent pas être utilisés. Notre conception utilise une stratégie simple pour réduire l'impact de la variabilité : la cellule binaire des matrices de mémoire est composée de deux memristors (structure 2T2R), et les bits sont stockés de manière complémentaire.

La Figure 3.2a illustre l'impact de la variabilité des dispositifs dans l'approche conventionnelle 1T1R. Dans ce cas, les memristors en état LRS signifient l'état zéro, et les memristors en état HRS signifient l'état un. Par conséquent, une erreur de bit se produit chaque fois qu'un dispositif censé être en LRS a une résistance supérieure au seuil entre LRS et HRS, ou qu'un dispositif censé être en HRS a une résistance inférieure à ce seuil. Dans l'approche 2T2R de la machine bayésienne (Figure 3.2b), inversement, deux memristors sont toujours programmés de manière complémentaire. L'état programmé est lu en comparant la résistance des deux memristors, à l'aide d'un amplificateur de détection avec précharge (PCSA : PreCharge Sense Amplifier). Dans ce cas, une erreur de bit se produit si un dispositif programmé en LRS a une résistance supérieure à celle de son dispositif complémentaire programmé en HRS. Par conséquent, les deux dispositifs doivent se trouver en fin de distribution pour qu'une erreur se produise, rendant les erreurs de bit beaucoup moins probables.

L'efficacité de l'approche 2T2R pour réduire les erreurs de bit est confirmée dans la Figure 3.2c. Cette figure trace le taux d'erreur de bit de l'approche 2T2R en fonction de celui de l'approche 1T1R, obtenu expérimentalement et théoriquement. Le résultat théorique suppose un amplificateur de détection parfait. Les résultats expérimentaux sont reproduits à partir de [8] et ont été obtenus sur un circuit intégré différent basé sur la même technologie que la machine bayésienne (la machine bayésienne est entièrement dédiée à l'approche 2T2R et ne permet pas d'estimer le taux d'erreur de bit de l'approche 1T1R). Nous avons également tracé le taux d'erreur qui serait obtenu en utilisant un code de correction d'erreur conventionnel à correction simple/détection double d'erreur (SECDED, ou Hamming étendu) qui double le nombre de memristors, comme notre approche. Nous constatons que notre approche réduit le nombre d'erreurs de bit presque de manière équivalente au code SECDED. Cependant, le code SECDED nécessite des circuits de décodage coûteux en surface, en délai et en énergie, tandis que, dans notre approche, la correction d'erreur se produit naturellement dans l'amplificateur de détection sans coût supplémentaire.

3.1.3 Étape de Programmation

Cette partie détaille la méthodologie de programmation des matrices de memristors dans la machine bayésienne fabriquée. Les memristors sont programmés avec des tensions supérieures à la tension nominale utilisée pour les circuits numériques (1.2 V). L'exigence de tension de programmation supérieure à celle nominale des memristors est un problème mineur dans les

systèmes qui séparent la mémoire du calcul, car un seul circuit haute tension dédié peut être associé à la matrice de mémoire. En revanche, la machine bayésienne comporte de multiples petites matrices de mémoire entièrement intégrées dans la logique. La programmation des memristors nécessite donc la distribution de tensions de programmation supérieures à la tension nominale localement et une stratégie de programmation appropriée. Gérer cette complexité a été le plus grand défi de conception pour obtenir une machine bayésienne fonctionnelle.

La machine bayésienne stocke des bits en utilisant une structure 2T2R comme nous l'avons vu précédemment (Fig. 3.3a) : la cellule de mémoire est composée d'un memristor "ligne de bit" et d'un memristor "ligne de bit inverse" (R et R_b), positionnés sur la même rangée et associés chacun à un transistor de sélection nMOSFET. Les deux memristors sont connectés à deux lignes de bit différentes (BL et BLb) du côté de leur électrode inférieure. Inversement, l'électrode supérieure des deux memristors est connectée à la même ligne source, afin de limiter le câblage et les circuits de programmation. Cette ligne source partagée nécessite une attention particulière lors de la programmation des memristors.

En plus de l'alimentation numérique (VDD), deux tensions d'alimentation supérieures à la tension nominale (VDDR et VDDC) sont distribuées à chaque matrice de mémoire.

L'opération de programmation est contrôlée par des signaux de tension nominale (CBL, CSL et CWL) et une adresse, tous fournis par l'unité de contrôle numérique de la machine bayésienne. Les décodeurs sélectionnent la rangée et la colonne adressées. Ensuite, des convertisseurs de tension locaux appliquent soit la masse soit des tensions supérieures à la tension nominale aux matrices de vraisemblance, lorsque cela est nécessaire:

- Chaque rangée de mémoire comprend un convertisseur de tension (LS, commandé par CWL), qui contrôle la ligne de mot (WL) qui alimente les grilles des transistors de sélection nMOS des rangées de mémoire. Selon la valeur de CWL, le convertisseur de tension de la rangée adressée connecte soit sa ligne de mot à la masse, soit à l'alimentation VDDR (voir Fig. 3.3b et e).

- De même, chaque colonne de mémoire comprend deux convertisseurs de tension (convertisseur de tension régulier LS, commandé par CSL, et convertisseur de tension à trois états TLS, commandé par CBL), qui contrôlent les lignes de source (SL) et les lignes de bit (BL). Ces convertisseurs de tension connectent la ligne de source et les deux lignes de bit soit à la masse, soit à l'alimentation VDDC. Sachant que les deux lignes de bit BL et BLb sont connectées au même convertisseur de tension, ils reçoivent donc toujours des tensions complémentaires (voir Fig. 3.3b et f).

Lorsque le système est d'abord caractérisé, les memristors doivent être formés. La Figure 3.3c montre les tensions qui doivent être appliquées sur les memristors lors de l'étape de forming. Le tableau de la Figure 3.3g répertorie les valeurs associées CSL et CBL. Lorsqu'un memristor est en cours de formation, l'autre memristor n'est pas affecté, car la ligne de bit et

la ligne de bit inverse voient toujours des tensions complémentaires (car elles sont connectées aux sorties complémentaires du convertisseur de tension BL). Une fois que les bonnes adresses, valeurs de CSL et CBL ont été définies, CWL est élevé à un, et la formation a lieu pendant l'impulsion CWL (d'une durée d'une microseconde, voir Fig. 3.3d).

Une fois que les memristors sont formés, ils sont tous dans un état de faible résistance. Les memristors peuvent alors être programmés et reprogrammés à volonté, soit en état de faible résistance (LRS) par une opération SET, soit en état de haute résistance (HRS) par une opération RESET.

Pour programmer une valeur 'zéro' dans la cellule mémoire, le memristor de la ligne de bit est programmé en haute résistance (RESET) et le memristor de la ligne de bit inverse en faible résistance (SET). L'inverse est fait pour programmer une valeur 'un' Fig. 3.3c montre les tensions requises pour programmer les valeurs un et zéro, et le tableau de Fig. 3.3g répertorie les valeurs correspondantes de CSL et CBL. Les niveaux de tension de VDDR et VDDC utilisés pour les opérations SET et RESET sont rappelés dans la Fig. 3.3d.

3.1.4 Étapes de Lecture et d'Inférence

Cette section explique l'opération de lecture des états des memristors dans la machine bayésienne fabriquée. Pendant le mode de lecture, les lignes de bits sont déconnectées du circuit de programmation (en utilisant le convertisseur de tension à trois états, voir Fig. 3.3b) et connectées au circuit de l'amplificateur de détection (PCSA). Les circuits de conversion du niveau de la source et de la ligne des mots agissent alors comme des tampons alimentés par la tension nominale VDD (voir Fig. 3.4a). L'opération de lecture fonctionne en deux phases : précharge et décharge (voir Fig. 3.4b).

L'amplificateur de détection (PCSA) est un amplificateur de détection économe en énergie qui fonctionne sans aucun chemin de courant direct entre la masse et l'alimentation, grâce à la phase de précharge initiale. Les détails du fonctionnement de ce circuit sont rapportés dans [96, 97], et nous les résumons ici. La précharge élève toutes les tensions de l'amplificateur de détection et des lignes de bits à l'alimentation numérique VDD (Fig. 3.4a et c). Dans l'opération de lecture réelle, les deux lignes de bits se déchargent (Fig. 3.4d,e). La ligne de bits avec la résistance la plus faible se décharge plus rapidement jusqu'à ce que sa sortie d'inverseur associée se décharge à la masse, ce qui verrouille la sortie d'inverseur complémentaire à la tension d'alimentation (Fig. 3.4f,g). Par conséquent, les deux tensions de sortie représentent la comparaison des deux valeurs de résistance complémentaires, ce qui donne le bit stocké dans la cellule de bits 2T2R.

Après cette opération de lecture, la machine bayésienne peut effectuer l'inférence bayésienne en utilisant le calcul stochastique. Les quatre registres à décalage à rétroaction linéaire (LFSR) générant des nombres pseudo-aléatoires à chaque cycle d'horloge sont situés dans l'unité de contrôle numérique, et leurs sorties sont distribuées aux matrices de mémoires de vraisemblance via des fils verticaux (voir section 2.2.1). Chaque bloc de vraisemblance utilise le circuit

présenté à la Figure 3.1f pour générer des bits aléatoires avec une probabilité égale à la valeur lue dans sa matrice de mémoire de vraisemblance, basée sur les sorties des LFSRs. Ce circuit donne des résultats équivalents à un comparateur, mais avec un coût de surface plus faible[7] (voir section 2.2.1). Les bits résultants sont combinés par des portes logiques AND, le tout se déroulant pendant un seul cycle d'horloge.

Pour comprendre le fonctionnement plus en détail, la Fig. 3.5a montre un schéma plus détaillé du bloc élémentaire de vraisemblance (à la ligne m et à la colonne n). La fonction de ce bloc est de produire un flux de bits avec une probabilité proportionnelle à la vraisemblance $P(O_n|Y_m)$. De plus, la Fig. 3.5b montre un diagramme de flux de l'inférence bayésienne stochastique dans la machine bayésienne, et la Fig. 3.5c montre un diagramme temporel du fonctionnement de la machine. Le code couleur tout au long de la Fig. 3.5 est le suivant : la couleur bleue correspond à la génération de nombres aléatoires, la couleur orange à la lecture de la mémoire, et la couleur verte à l'inférence stochastique proprement dite.

Avant de commencer les opérations d'inférence bayésienne, nous devons d'abord charger les seeds des LFSR. Dans notre conception, les seeds d'entrée sont chargées à partir des pads d'entrée et acheminées vers les quatre LFSR de la machine bayésienne par l'unité de contrôle numérique de la machine bayésienne. Pouvoir choisir la seed du LFSR est important pour notre étude, comme nous allons le voir dans la section suivante. Dans une conception finale, les valeurs optimales des seeds du LFSR pourraient être chargées automatiquement par l'unité de contrôle.

Comme les LFSR ont une sortie périodique, l'initialisation de la seed doit être effectuée une seule fois tant que l'alimentation numérique VDD reste allumée. La machine bayésienne effectue ensuite l'inférence en deux phases principales :

- **Lecture de la mémoire.** Les valeurs de vraisemblance sont lues à partir des tableaux de memristors, en fonction des observations d'entrée (qui agissent comme des adresses de ligne). Les observations $O_1, ..., O_n$ sont des entrées hors puce chargées à partir de pads d'entrée dédiés, puis adressées aux tableaux de mémoire de vraisemblance par l'unité de contrôle numérique de la machine bayésienne (une observation pour chaque colonne). Tous les tableaux de mémoire de vraisemblance sont lus simultanément.

- **Phase d'inférence stochastique itérative.** À chaque cycle d'horloge, les LFSR génèrent de nouveaux nombres pseudo-aléatoires de huit bits. Ces nombres alimentent les circuits comparateurs pour calculer les bits stochastiques en fonction des valeurs de vraisemblance lues. Les sorties des circuits comparateurs de la même ligne sont alimentées à une chaîne de portes ET, pour effectuer les multiplications stochastiques ; les résultats de ces multiplications représentent les sorties de la machine bayésienne. Toutes ces opérations sont effectuées par des circuits purement combinatoires en un cycle d'horloge. Comme la périodicité des LFSR est de 255 cycles, le nombre maximum d'itérations est de 255 cycles.

3.1.5 Synthèse : Étapes pour l'Utilisation de la Machine Bayésienne

En résumé, la Fig. 3.6 récapitule également les différentes étapes d'un projet utilisant la machine bayésienne, de l'entraînement à l'opération sur puce :

- **Collecter les données d'entraînement.** Tout projet commence par la collecte de données d'entraînement.
- **Mettre en œuvre le modèle de vraisemblance.** Les vraisemblances des modèles bayésiens sont calculées. Dans notre tâche d'échantillonnage de reconnaissance de gestes, les vraisemblances sont modélisées en ajustant les lois gaussiennes sur les données d'entraînement. Dans d'autres situations, les modèles de vraisemblance peuvent également être obtenus sur la base de connaissances d'experts ou d'informations préalables [98].
- **Normaliser et quantifier les vraisemblances.** Pour améliorer l'efficacité du calcul stochastique, les vraisemblances sont normalisées par colonne par la valeur de vraisemblance maximale de la colonne. Les vraisemblances sont ensuite discrétisées en entiers de huit bits, et les modèles sont quantifiés au nombre de valeurs d'observation prises en charge par la machine bayésienne.
- **Programmer la machine bayésienne.** Les valeurs de vraisemblance sont programmées dans les memristors de la machine bayésienne en suivant la méthodologie décrite précédemment.
- **Utiliser la machine bayésienne pour faire de l'inférence.** La machine bayésienne peut enfin être utilisée pour inférer des variables sur la base d'observations, en utilisant la méthodologie décrite dans cette partie.

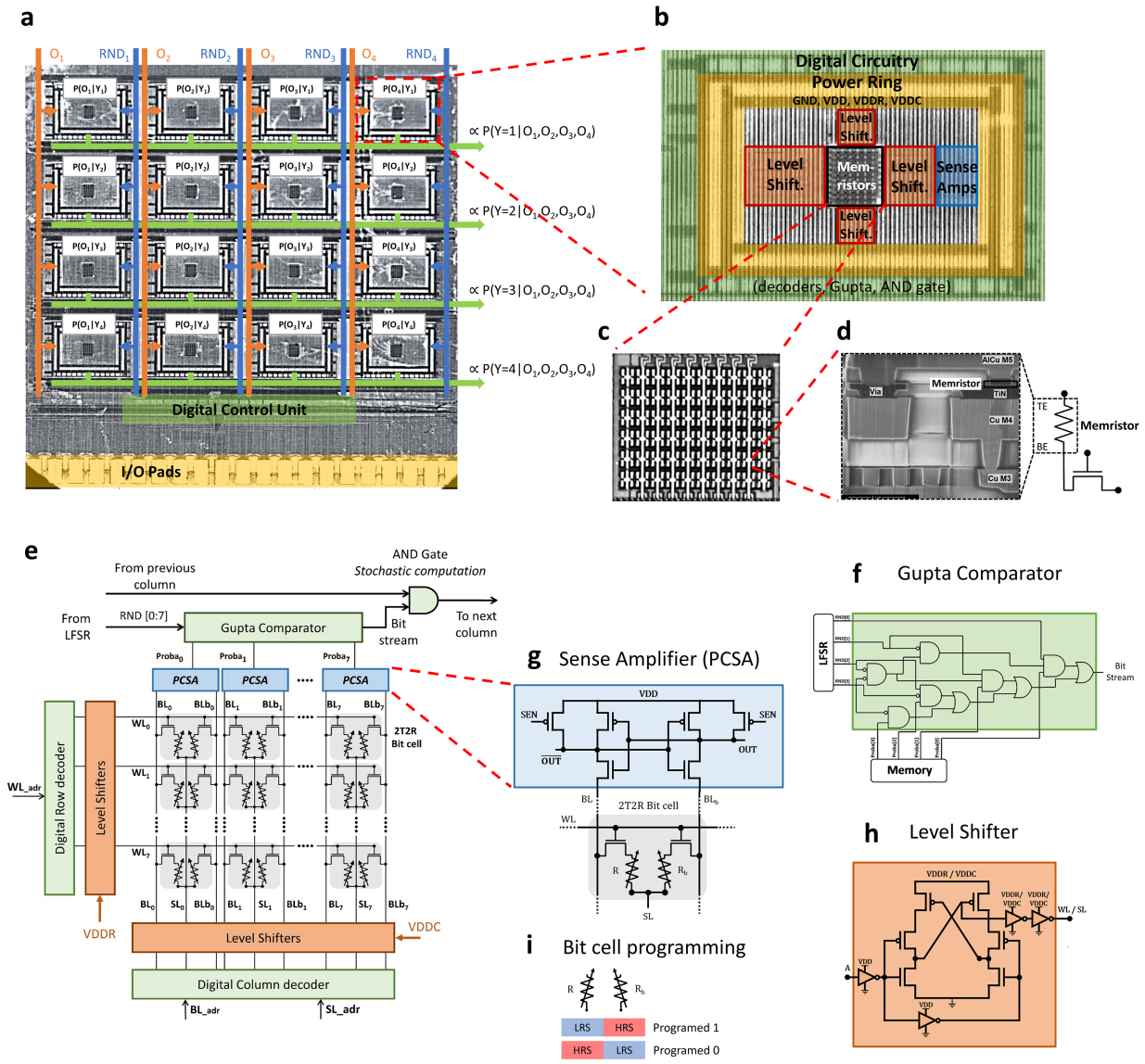


Figure 3.1: **Machine bayésienne à base de memristors fabriquée.** **a** Photographie au microscope optique de la puce du système bayésien. **b** Détail du bloc de vraisemblance, qui comprend un circuit numérique et un bloc de mémoire avec son circuit périphérique. **c** Photographie du réseau de memristors 2T2R. **d** Image de microscopie électronique à balayage d'un memristor dans le back-end de notre processus hybride memristor/CMOS. **e** Schéma du bloc de vraisemblance présenté dans **b**. **f** *Schéma du circuit comparateur Gupta*[7], utilisé pour générer le flux de bits aléatoires de vraisemblance en comparant la probabilité lue en mémoire avec le nombre aléatoire généré par le LFSR. Pour des raisons de compacité, le circuit est présenté ici dans une version à quatre bits. Une version à *huit bits* est mise en œuvre sur la puce. **g** Schéma de l'amplificateur de détection différentiel utilisé pour lire les états binaires des memristors. **h** Schéma du convertisseur de tension, utilisé pour adapter la tension d'entrée nominale aux tensions de formation et de programmation des memristors. **i** Principe de la programmation complémentaire des memristors de la cellule de bit 2T2R. Toutes les sous-figures utilisent des codes de couleur cohérents.

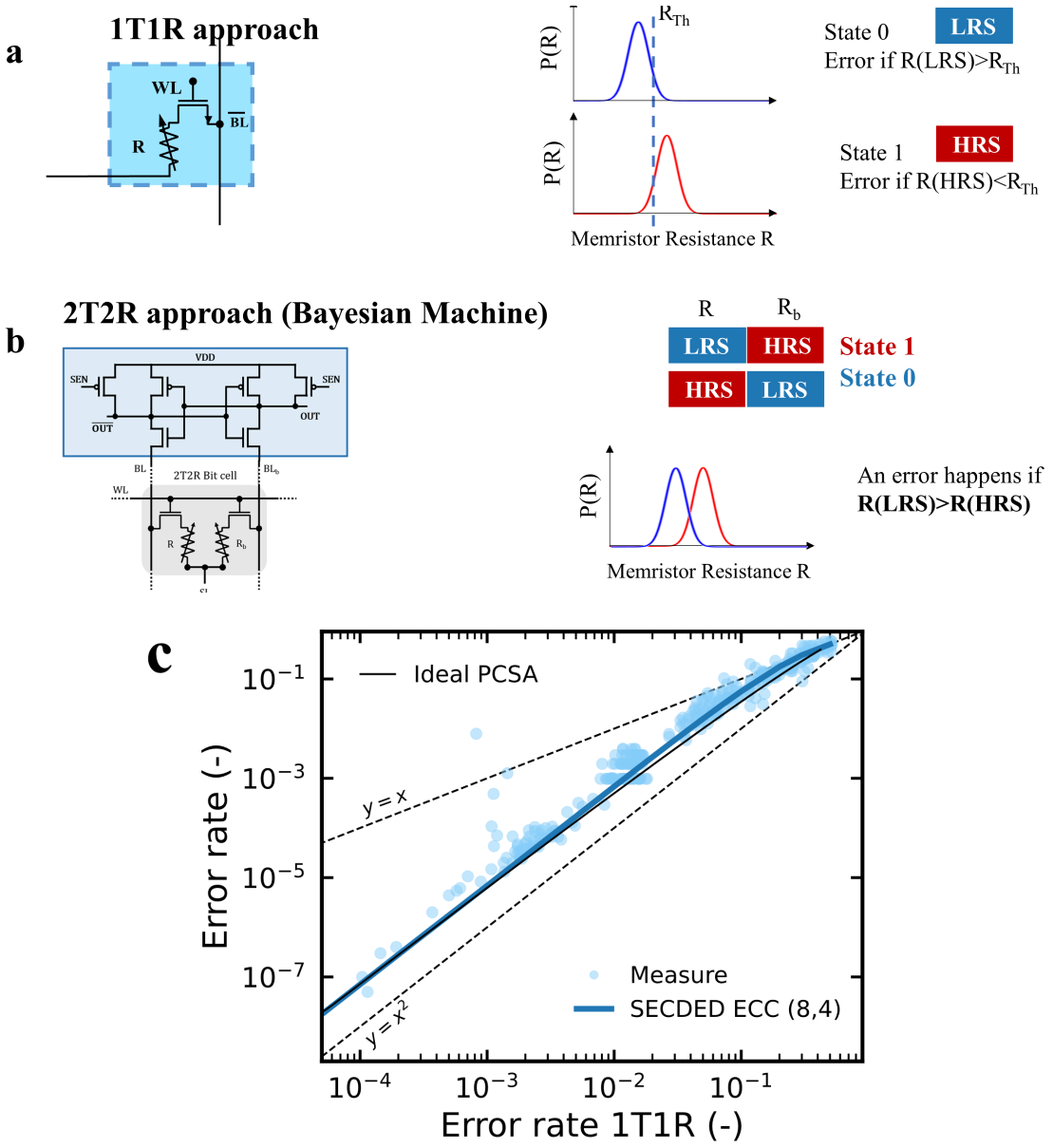
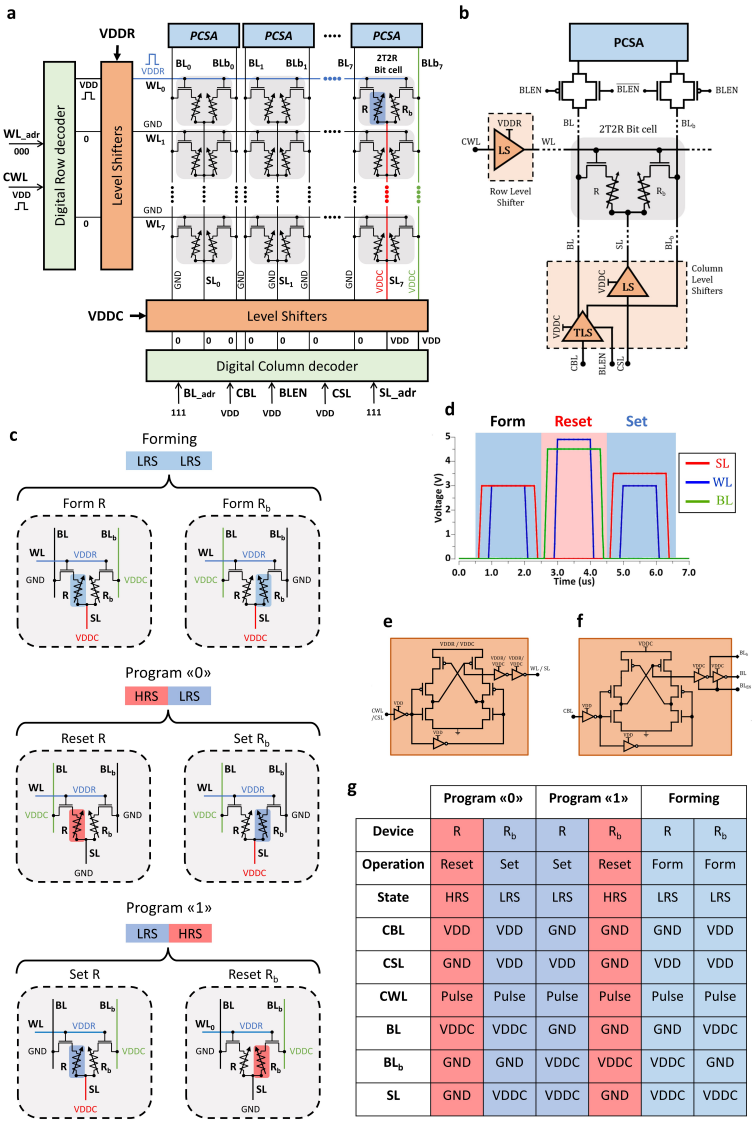


Figure 3.2: **Réduction des effets de la variabilité des dispositifs avec l'approche 2T2R**. **a** Dans l'approche conventionnelle, une erreur se produit si un dispositif programmé en LRS a une résistance supérieure au seuil entre LRS et HRS, ou si un dispositif programmé en HRS a une résistance inférieure à ce seuil. **b** Dans l'approche 2T2R de la machine bayésienne, une erreur se produit si un dispositif programmé en LRS a une résistance supérieure à celle de son dispositif complémentaire programmé en HRS : les deux dispositifs doivent se trouver en fin de distribution pour qu'une erreur se produise. **c** Taux d'erreur de l'approche 2T2R en fonction du taux d'erreur de l'approche 1T1R, dans des simulations supposant un amplificateur de détection parfait (ligne noire) ou mesuré expérimentalement sur le circuit intégré de [8] (points bleu clair). Ligne bleue : taux d'erreur d'un code ECC SECDED utilisant le même nombre de dispositifs que notre approche 2T2R.



	Program «0»		Program «1»		Forming	
Device	R	R_b	R	R_b	R	R_b
Operation	Reset	Set	Set	Reset	Form	Form
State	HRS	LRS	LRS	HRS	LRS	LRS
CBL	VDD	VDD	GND	GND	GND	VDD
CSL	GND	VDD	VDD	GND	VDD	VDD
CWL	Pulse	Pulse	Pulse	Pulse	Pulse	Pulse
BL	VDDC	VDDC	GND	GND	GND	VDDC
BL_b	GND	GND	VDDC	VDDC	VDDC	GND
SL	GND	VDDC	VDDC	GND	VDDC	VDDC

Figure 3.3: **Circuit de programmation pour les matrices de mémoire de vraisemblance.** **a** Schémas détaillés de la matrice de mémoire de vraisemblance, avec son circuit de périphérie de programmation et de lecture, affichant les tensions nécessaires pour effectuer une opération SET sur le premier memristor R de la première rangée, dernière colonne. **b** Schémas des connexions de la cellule de bit 2T2R au circuit de lecture et de programmation. Deux convertisseurs de tension (convertisseur de tension conventionnel LS et convertisseur de tension à trois états TLS) et un amplificateur de détection (PCSA) sont implémentés dans chaque colonne. Un convertisseur de tension est implémenté dans chaque rangée. Le signal numérique BLEN permet de choisir entre le mode de lecture ou de programmation. **c** Tensions qui doivent être appliquées sur la ligne de bit BL, la ligne de bit inverse BLb et la ligne source SL pour le forming, la programmation d'un 'zéro' et la programmation d'un 'un' dans une cellule de bit 2T2R. **d** Niveaux de tension et chronogrammes utilisés pour les opérations de forming, de RESET et de SET. **e** Schémas au niveau du transistor du convertisseur de tension (LS) et **f** des circuits du convertisseur de tension à trois états (TLS). **g** Tableau résumant la configuration des signaux de programmation (convertisseurs de tension) pour les différentes opérations de programmation prises en charge par la matrice de mémoire (forming, programmation d'un 'zéro' et programmation d'un 'un').

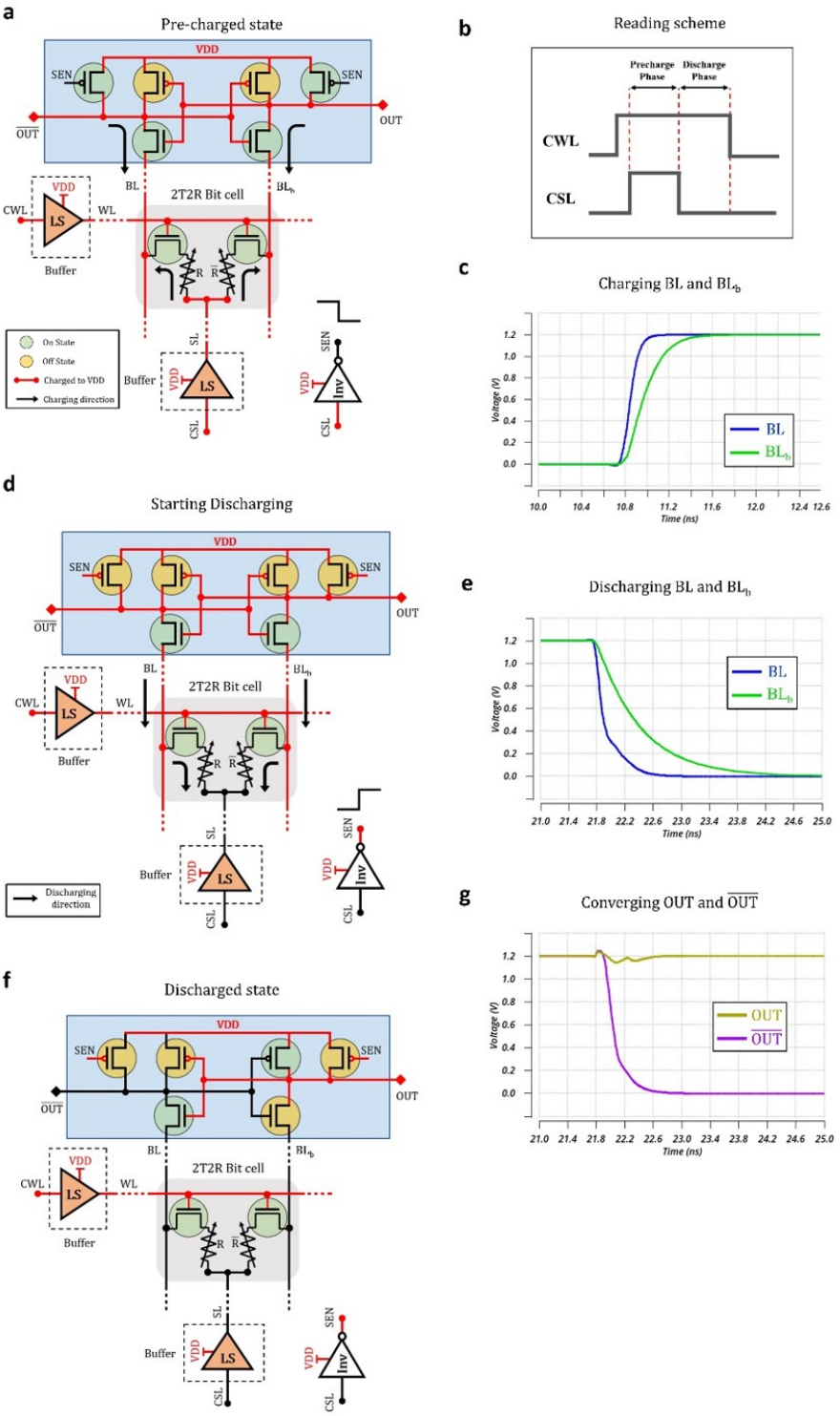


Figure 3.4: **Circuit de lecture pour les matrices de mémoire de vraisemblance.** Cette figure présente les schémas de circuit (**a**, **d** et **f**) et les simulations (**c**, **e**, **g** et **h**) qui expliquent l'opération de lecture de la matrice mémoire. **b** Le schéma de lecture implique une phase de précharge et une phase de décharge. **a** Schéma et **c** simulation du circuit de la phase de précharge. BL et BLb sont chargées à VDD. **d** Schéma et **e** simulation du circuit de la phase de décharge, BL et BLb sont déchargées à la masse avec une vitesse différente. **f** Schémas et **g** simulation des sorties de l'amplificateur de détection convergeant vers un état stable, pendant que BL et BLb sont complètement déchargées à la masse.

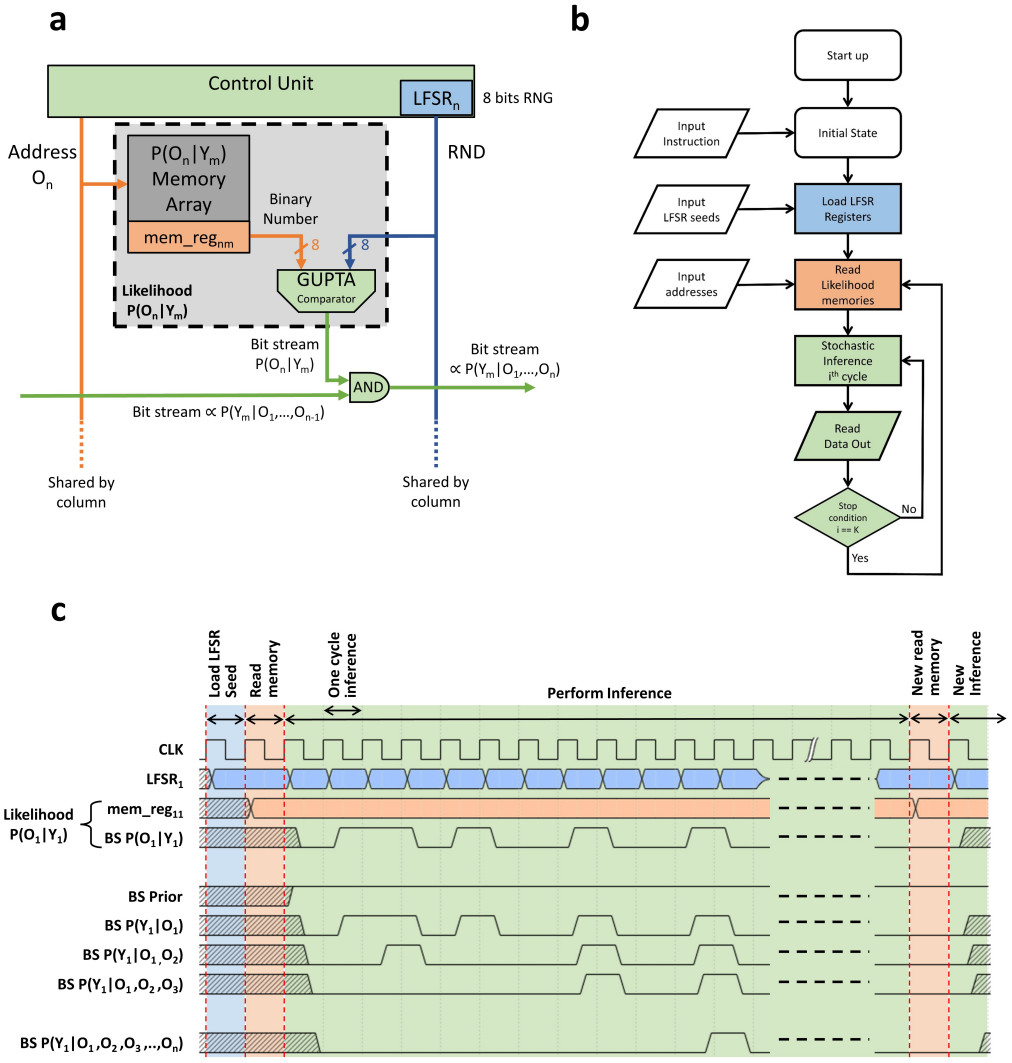


Figure 3.5: **Opération détaillée de la machine bayésienne.** **a** Schéma illustrant l'architecture détaillée d'un bloc élémentaire de vraisemblance. **b** Diagramme de flux des différentes opérations pour effectuer un calcul d'inférence bayésienne dans la machine bayésienne. **c** Diagramme temporel illustrant le fonctionnement de la machine bayésienne.

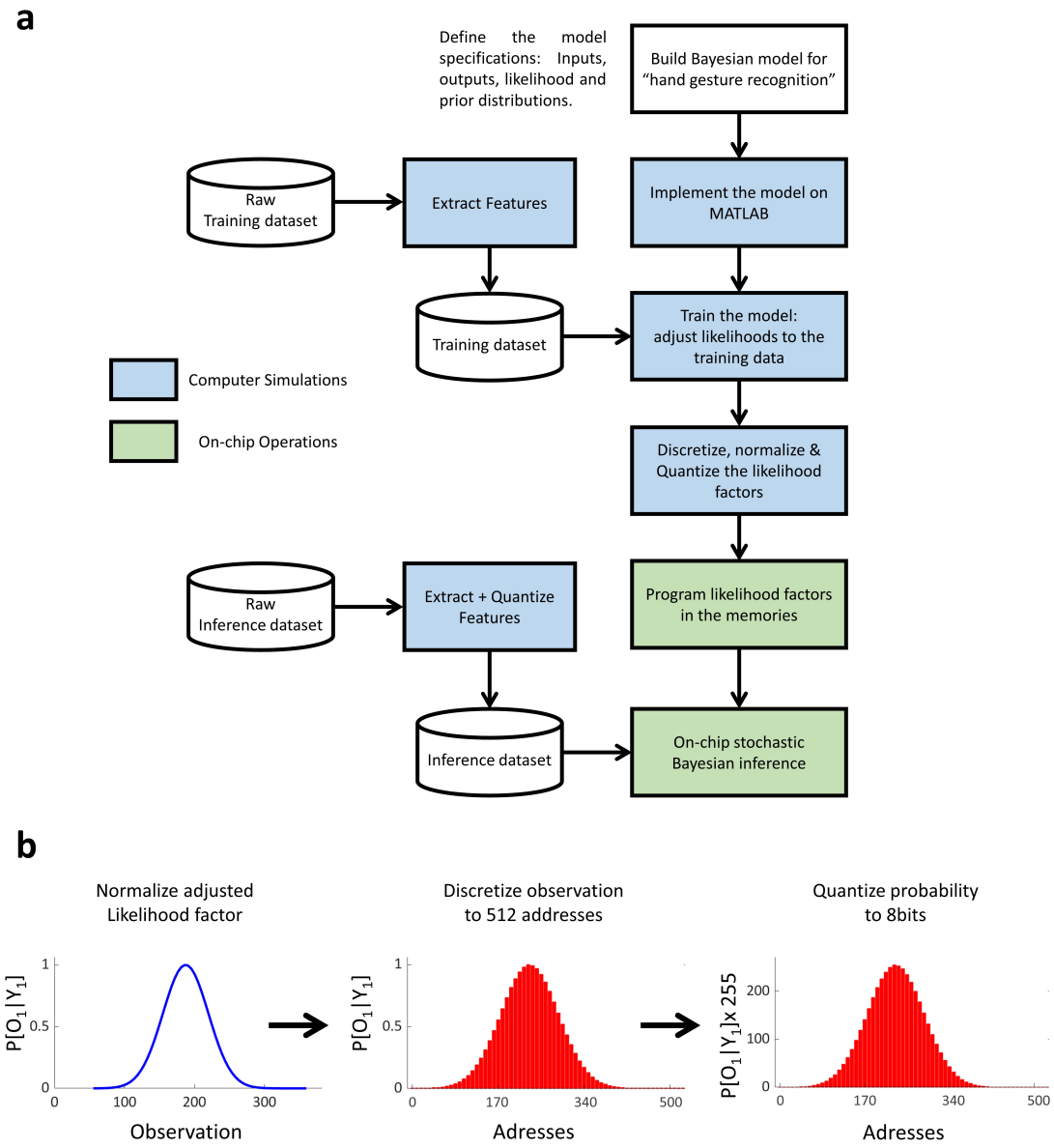


Figure 3.6: **Les différentes étapes d'un projet avec la machine bayésienne, de l'entraînement à l'inférence sur puce.** **a** Diagramme résumant les principales étapes d'un projet impliquant la machine bayésienne, de la construction du modèle bayésien à l'utilisation de la machine bayésienne pour effectuer une inférence sur puce. **b** Illustration des différentes étapes de normalisation, de discrétisation et de quantification des facteurs de vraisemblance, nécessaires avant la programmation des tableaux de mémoire de vraisemblance sur puce.

3.2 Analyse Approfondie : Mesures et Caractérisation

Notre système est testé à l'aide d'une carte à pointes personnalisée comprenant 25 pointes, connectée via des connecteurs SubMiniature A (SMA) à une carte de circuit imprimé (PCB) dédiée sur mesure comportant des convertisseurs de tension et d'autres éléments discrets. Le PCB relie les entrées et sorties de notre puce de test à un microcontrôleur ST Microelectronics STM32F746ZGT6, deux générateurs de forme d'onde et de mesure rapide Keysight B1530A, et un oscilloscope Tektronix DPO 3014. Le microcontrôleur est connecté à un ordinateur via une connexion série, tandis que les autres équipements sont connectés à l'ordinateur via une connexion GPIB National Instruments. Les tests sont ensuite effectués en utilisant Python dans un seul bloc-notes Jupyter contrôlant l'ensemble de la configuration.

Comme il a été noté dans la partie 3.1.3, sur la programmation des memristors, avant le fonctionnement de la machine bayésienne, les memristors doivent subir une opération unique de forming pour créer des filaments conducteurs. Cette étape est réalisée memristor par memristor avec les conditions suivantes : VDDC est réglé à 3,0 volts, VDDR à 3,0 volts, et VDD à 1,2 volts. Une fois formés, les memristors peuvent être programmés dans des états de faible résistance (LRS) ou de haute résistance (HRS) avec les conditions suivantes : pour la programmation en LRS, le mode SET est activé. VDDC est réglé à 3,5 volts, VDDR à 3,0 volts, et VDD à 1,2 volts. Pour la programmation en HRS, un mode RESET est activé. VDDC est réglé à 4,5 volts, VDDR à 4,9 volts, et VDD à 1,2 volts.

Une fois que les vraisemblances ont été programmées, des tensions élevées ne sont plus nécessaires, et la puce de test peut être utilisée dans son mode normal pour effectuer une inférence bayésienne basée sur les observations. Les trois tensions d'alimentation sont réglées sur la tension d'alimentation d'inférence, normalement de 1,2 volts, mais qui peut être réduite jusqu'à 0,5 volts. Les sorties de la machine bayésienne sont récupérées par l'unité microcontrôleur et transmises à l'ordinateur.

3.2.1 Exploration des Applications à Faible Tension : La Puce sans Package

Pour nos expériences, nous avons programmé les motifs présentés dans la Figure 3.7b dans les matrices de mémoire de vraisemblance. Ces motifs artificiels ont été construits de manière à ce que l'ensemble des inférences permettent d'explorer la totalité de la plage possible de probabilités en sortie (Fig. 3.8a, b), ce qui permet de tester la fonctionnalité du démonstrateur. La figure 3.7a représente quant a elle les valeurs des memristors avant la première étape de forming, comme les deux memristors sont dans un état de haute résistance, l'amplificateur de détection attribue de manière aléatoire la valeur "0" ou "1".

Les memristors sont toujours programmés de manière complémentaire, suivant le principe établi dans [8] et décrit précédemment. Aucune stratégie de programmation et de vérification

n'est employée. Les vraisemblances sont stockées sous forme d'entiers de huit bits proportionnels aux valeurs de vraisemblance (voir Fig. 3.7b). Elles sont normalisées par la valeur maximale des vraisemblances d'une colonne (c'est-à-dire que la valeur de vraisemblance maximale est stockée sous forme d'entier FF, en représentation hexadécimale). Cette normalisation permet au calcul stochastique de converger plus rapidement.

La Figure 3.7c montre les vraisemblances effectivement programmées dans le démonstrateur, telles que mesurées juste après l'opération de programmation (voir section 3.1.3). Aucune erreur n'est observée par rapport au motif prévu, mettant en évidence les performances de notre approche complémentaire pour programmer les memristors. La Figure 3.7d montre les mesures des vraisemblances stockées cinq mois après la programmation (le démonstrateur a été conservé à température ambiante pendant ces cinq mois). Aucune erreur n'est observée.

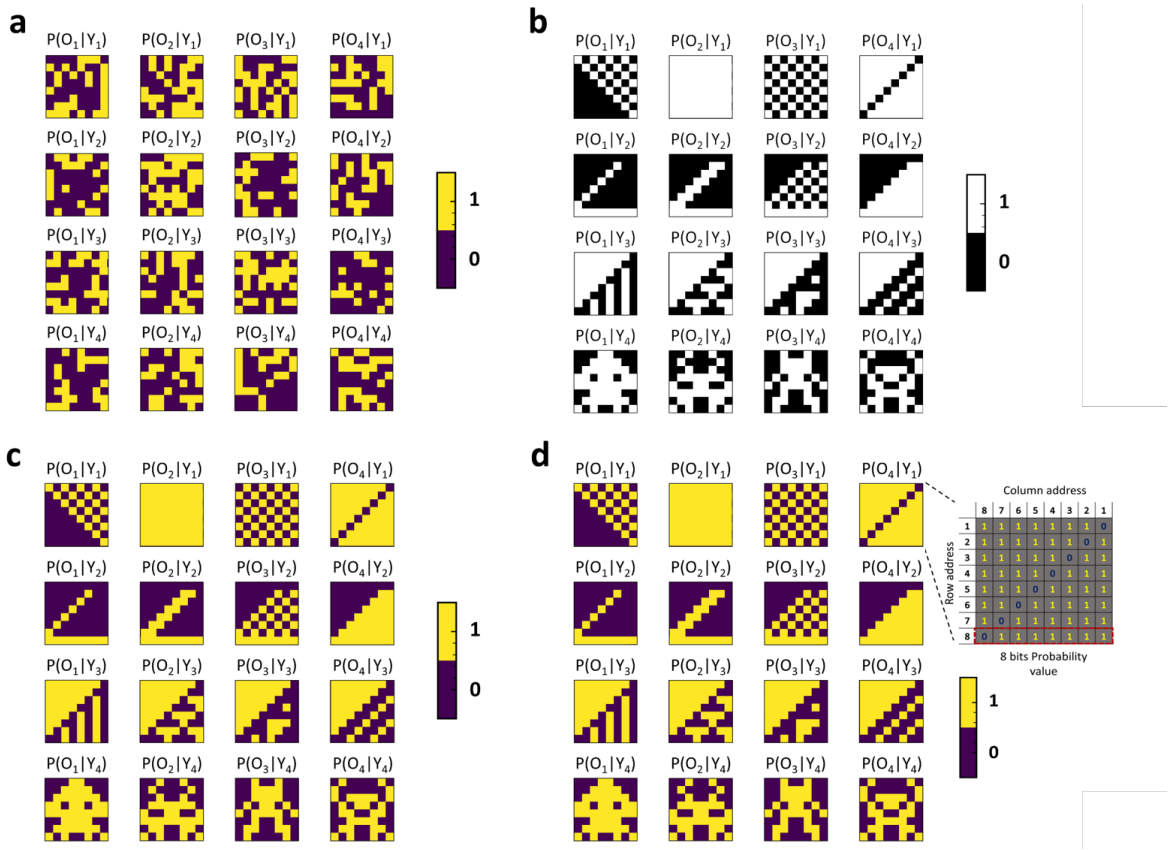


Figure 3.7: **Mesures des vraisemblances stockées dans le démonstrateur.** **a** Motifs mesurés avant l'étape de formation (forming). Les résultats des mesures semblent aléatoires. **b** Motifs destinés à être stockés. **c** Motifs mesurés immédiatement après la programmation. **d** Motifs mesurés après cinq mois, pendant lesquels le démonstrateur a été conservé à température ambiante. Les images **a**, **c**, **d** sont présentées en couleurs pour exprimer qu'elles représentent des mesures effectuées sur puce, tandis que l'image **b** est présentée en noir et blanc pour exprimer qu'elle représente un motif prévu.

Le fonctionnement réel de la machine bayésienne est présenté dans la Figure 3.8a-b. Dans

la Figure 3.8a, les LFSRs ont été initialisés à l'aide de seeds aléatoires. L'axe des abscisses représente le résultat théorique attendu de la loi de Bayes (c'est-à-dire la sortie souhaitée pour la machine bayésienne), tandis que l'axe des ordonnées représente les résultats mesurés expérimentalement en comptant les bits en sortie de la puce. Les différents points de la Figure 3.8 sont obtenus en changeant aléatoirement les entrées O_1, O_2, O_3 et O_4 du circuit, ainsi que la tension nominale d'alimentation V_{DD}. Pour chaque ensemble d'entrées aléatoires, le système a été exploité pendant 255 cycles d'horloge (c'est-à-dire la périodicité des LFSRs). Le nombre de "1" en sortie de chaque rangée est compté et divisé par 255 pour être converti en probabilité, et représenté dans la Figure 3.8.

Dans la Figure 3.8a, on constate que les probabilités mesurées suivent de près la loi de Bayes, avec quelques écarts. Ces écarts peuvent être attribués à la nature imparfaite des nombres pseudo-aléatoires générés par les LFSRs. Les nombres générés sur les différentes colonnes présentent des corrélations, ce qui empêche le calcul stochastique d'être parfaitement précis .

Heureusement, cette imperfection peut être évitée par un choix intelligent des seeds des LFSRs comme nous l'avons vu dans la section 2.2.2. Dans les mesures de la Figure 3.8b, la puce a été initialement programmée en utilisant un choix optimal de seeds. On constate que les mesures suivent parfaitement la loi de Bayes pour toutes les entrées possibles, ce qui met en évidence le fort potentiel du calcul stochastique pour l'inférence bayésienne.

Enfin, en raison de sa nature entièrement numérique, le système est flexible en termes de tension d'alimentation. Les Figures 3.8a-b montrent que le système reste pleinement fonctionnel lorsque la tension d'alimentation est réduite jusqu'à une valeur de 0.6volts, bien que la tension nominale de notre technologie CMOS soit de 1.2volts. Cette opération permet de réduire la consommation d'énergie d'un facteur d'environ quatre. À des tensions plus basses (points bleu clair), l'inférence bayésienne devient moins précise. Cette limite de tension est due à la valeur de la tension de seuil des transistors à oxyde épais utilisés dans la matrice de mémoire, d'environ 0.6 volts. Des tensions d'alimentation encore plus basses pourraient donc être utilisées en utilisant des transistors à seuil de tension plus basse.

3.2.2 La Puce Packagée : Comprendre les Perturbations de Lecture

Pour la deuxième partie des tests de la puce, la machine bayésienne a été mise dans un boîtier de type JLCC52. Un des avantages d'avoir une puce dans un boîtier est la facilité de manipulation mais également une protection physique. Nous avons donc intégré la puce dans son boîtier dans un PCB associé encore une fois à un microcontrôleur STM32 relié en série à un ordinateur. Pour les autres équipements, nous utilisons une alimentation stabilisée (KEITHLEY-2230G-30-1) pour les tensions VDD, VDDR et VDDC. Nous n'avons plus besoin d'autres équipements : tout est fait sur le PCB. Les mesures sont faites dans les mêmes conditions décrites précédemment. La Figure 3.9 montre le setup de test comprenant tous les éléments nécessaires. L'utilisation de cette puce dans un boîtier nous permet de réaliser des mesures plus poussées notamment en terme de durée d'utilisation comme faire un grand nombre de lectures successives pour

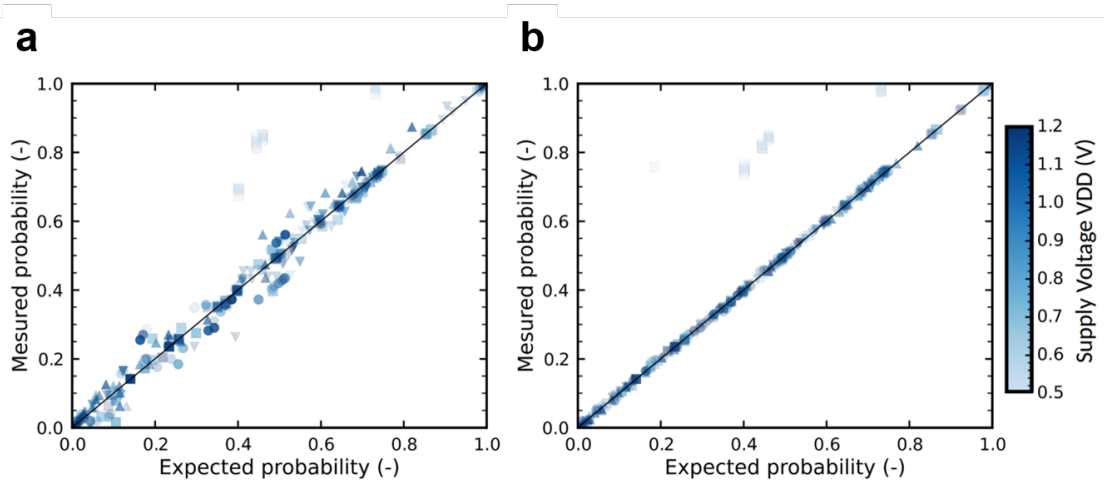


Figure 3.8: **Mesures de la machine bayésienne à base de memristors fabriquée.** **a** Sortie de la machine bayésienne : probabilité postérieure mesurée en fonction de la valeur attendue de la loi de Bayes. Les différents points correspondent à des entrées d'observations aléatoires. Les différentes lignes sont regroupées dans le même graphique. Les points sont obtenus avec différentes tensions d'alimentation VDD comprises entre 0,5 et 1,2 volts. Ce graphique est obtenu avec des *seeds* LFSR non optimaux. Les probabilités mesurées sont obtenues en moyennant les mesures expérimentales sur toute la période LFSR (255 cycles). **b** Identique à **a**, en utilisant des *seeds* LFSR optimaux. Les symboles indiquent quelle ligne de la machine bayésienne a été utilisée (cercle, triangle vers le haut, triangle vers le bas, carré : première, deuxième, troisième et quatrième ligne).

observer les effets de la perturbation de lecture.

En effet, le fonctionnement typique d'une machine bayésienne nécessite des opérations de lecture fréquentes sur les matrices de vraisemblance, avec des reprogrammations rares : les memristors doivent être reprogrammés uniquement lorsque le modèle est modifié (par exemple, après un recalibrage basé sur de nouvelles données d'entraînement). Il est donc essentiel que les opérations de lecture ne puissent pas changer accidentellement l'état des dispositifs de mémoire (effet de perturbation de lecture). Pour évaluer l'existence de l'effet de perturbation de lecture, nous avons effectué des opérations de lecture répétées sur un bloc de mémoire de vraisemblance de la puce fabriquée. Les opérations de lecture sont effectuées par les amplificateurs de détection intégrés (voir section 3.1.4). Nous avons utilisé une tension d'alimentation numérique de 1,2 volts, car c'est la tension d'alimentation numérique la plus élevée prise en charge par notre système, et donc plus susceptible de causer des effets de perturbation de lecture. Notre configuration expérimentale permet de lire la matrice de vraisemblance environ un million de fois par jour. Nous avons observé qu'après cinq jours d'opérations de lecture continues, soit un total de 5,7 millions d'opérations de lecture d'une matrice de mémoire de vraisemblance complète, aucun bit de vraisemblance n'avait changé (voir Fig. 3.10), suggérant une forte immunité aux effets de perturbation de lecture.

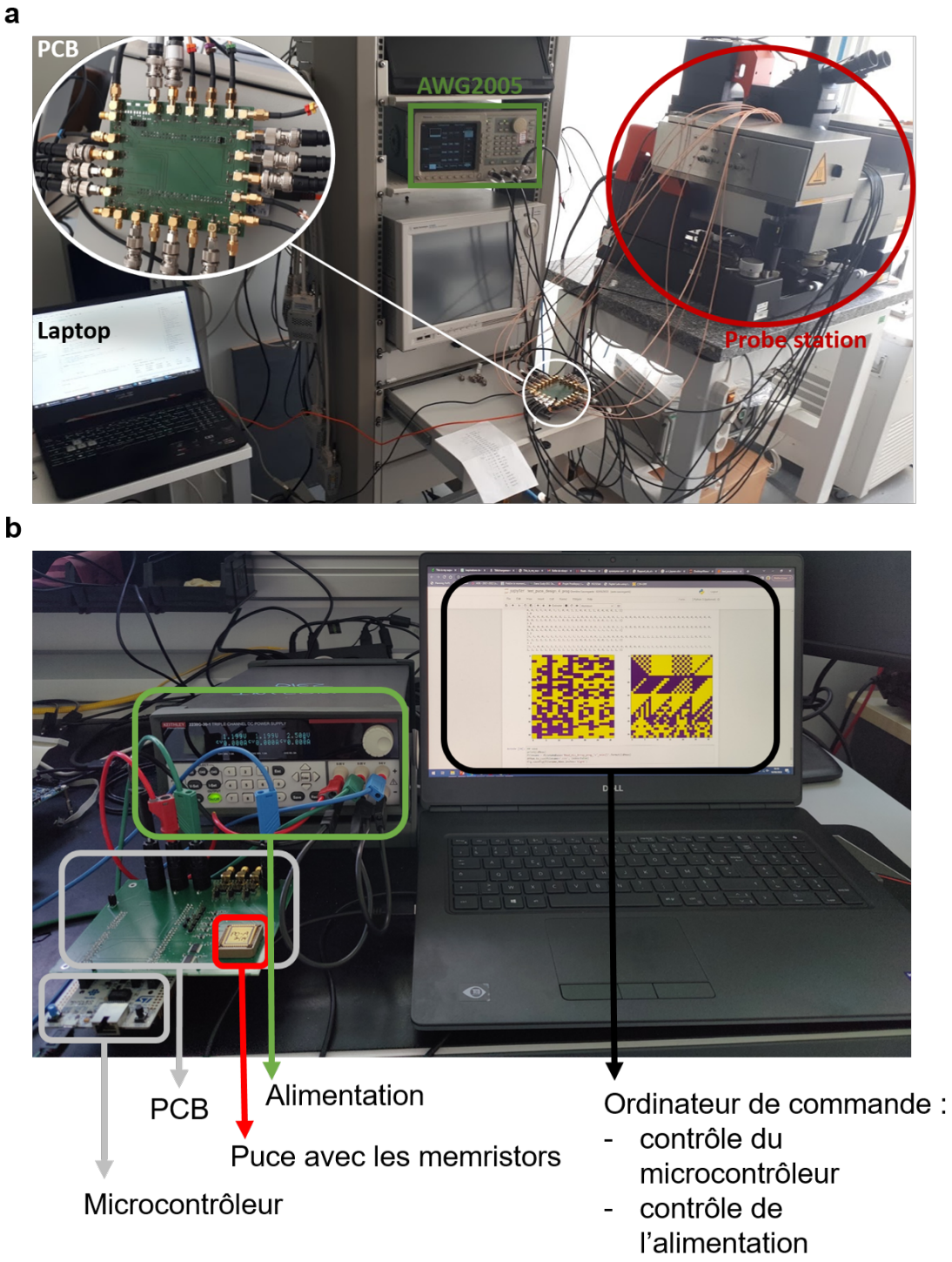


Figure 3.9: **Systèmes de tests des deux puces bayésiennes.** **a** Installation sous la station sous pointes au laboratoire IM2NP à Marseille et **b** Installation dans un package JLCC52 et PCB, réalisée au C2N.

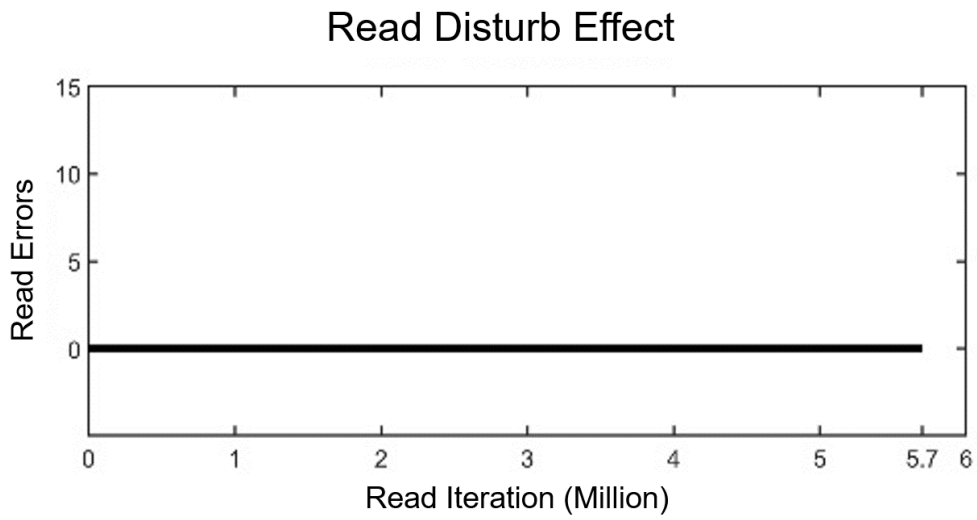


Figure 3.10: **Mesure expérimentale de la perturbation due à la lecture** sur une matrice de mémoire de vraisemblance, avec une valeur VDD de 1,2 volts. Même après 5,7 millions d'opérations de lecture de l'ensemble de la matrice, aucune erreur n'est observée.

L'immunité aux effets de perturbation de lecture de notre machine peut s'expliquer par trois raisons principales :

- Les memristors à base d'oxyde d'hafnium sont naturellement résistants aux effets de perturbation de lecture en raison de la nature hautement non linéaire de leur processus de commutation [99].

- L'approche complémentaire 2T2R réduit non seulement l'impact de la variabilité des dispositifs, mais aussi les effets de perturbation de lecture. Même si l'effet de perturbation de lecture augmente la résistance d'un dispositif à faible résistance, la vraisemblance stockée serait affectée uniquement si le dispositif perturbé finit par avoir une résistance supérieure à son dispositif complémentaire à haute résistance.

- L'amplificateur de détection utilisé pour lire les dispositifs atténue naturellement les effets de perturbation de lecture. Lorsque l'amplificateur de détection a identifié le bit stocké, les nœuds reliant l'amplificateur de détection à la matrice de memristors sont rapidement ramenés à la masse, et les memristors de lecture voient donc une tension nulle. Par conséquent, le courant est appliqué aux memristors uniquement pendant le temps nécessaire à l'amplificateur de détection pour différencier les deux états de mémoire possibles. Ce mode de fonctionnement contraste avec les amplificateurs de sensibilité en mode courant conventionnels où le courant est appliqué pendant un temps fixe qui doit être choisi dans un scénario de pire cas [100].

3.3 Estimations des Performances Énergétiques

Notre puce de test permet de valider la faisabilité des défis liés à la conception et à la fabrication de la machine bayésienne à base de memristors. Cependant, ce système n'est pas adapté pour évaluer la consommation d'énergie d'un système final, car la puce de test est trop petite pour mettre en œuvre des applications réelles. De plus, les contraintes du processus semi-académique et l'utilisation de transistors larges entraînent une forte augmentation de la consommation d'énergie capacitive dynamique.

Pour évaluer la consommation d'énergie, nous passons à une conception plus grande et à une application réaliste (la reconnaissance de gestes étudiée dans le chapitre 2), et utilisons des outils de conception de circuits intégrés standard de l'industrie pour évaluer la consommation d'énergie avec une échelle fine. Ces outils nous permettent d'obtenir une évaluation plus précise de la consommation d'énergie pour une application concrète.

Les estimations énergétiques sur la conception mise à l'échelle ont été obtenues à l'aide d'une méthodologie hybride. Ces estimations se concentrent sur la phase d'inférence, c'est-à-dire l'utilisation réelle de la machine bayésienne lorsque diverses entrées sont présentées, après que les memristors aient été formés et que les vraisemblances aient été programmées.

La consommation énergétique des matrices de memristors elles-mêmes est obtenue à l'aide de simulations de circuits (basées sur le simulateur Siemens Eldo), incluant les capacités parasites extraite de la disposition des matrices de mémoire.

La consommation d'énergie du reste du système est obtenue à l'aide du framework Cadence Voltus pour l'intégrité de puissance. Ces estimations utilisent des fichiers de modification de valeurs (VCD) obtenus à partir de notre banc de test, ce qui garantit que les estimations énergétiques correspondent à une situation réaliste.

3.3.1 Simulations : Un Outil pour l'Analyse Énergétique

Un défi de ces estimations réside dans le fait qu'il est crucial que Cadence Voltus modélise correctement le comportement des matrices de memristors, car la consommation d'énergie du système dépend directement de la sortie de ces blocs. Cependant, étant des blocs personnalisés et non inclus dans la bibliothèque standard de la fonderie, des développements spéciaux ont été nécessaires. Nous avons programmé, à l'aide de MATLAB, un compilateur de fichiers liberty pour les matrices de memristors, fournissant à Cadence Voltus une description de la fonctionnalité de la matrice en fonction des vraisemblances programmées dans un bloc de mémoire. Lors de la simulation MATLAB, nous extrayons les valeurs intermédiaires de l'inférence et les incluons en tant que sortie de mémoire. À cette fin, nous créons un nouveau fichier liberty qui sera utilisé dans l'opération de placement et routage. Ce fichier liberty spécifie la sortie pour chaque mémoire en fonction de l'entrée et des adresses. Cette méthode ne peut être utilisée que pour estimer la consommation d'énergie pendant la phase d'inférence, car les sorties

sont spécialement adaptées à cette phase. Le schéma modélisant les étapes nécessaires à la simulation est donné par la figure 3.11.

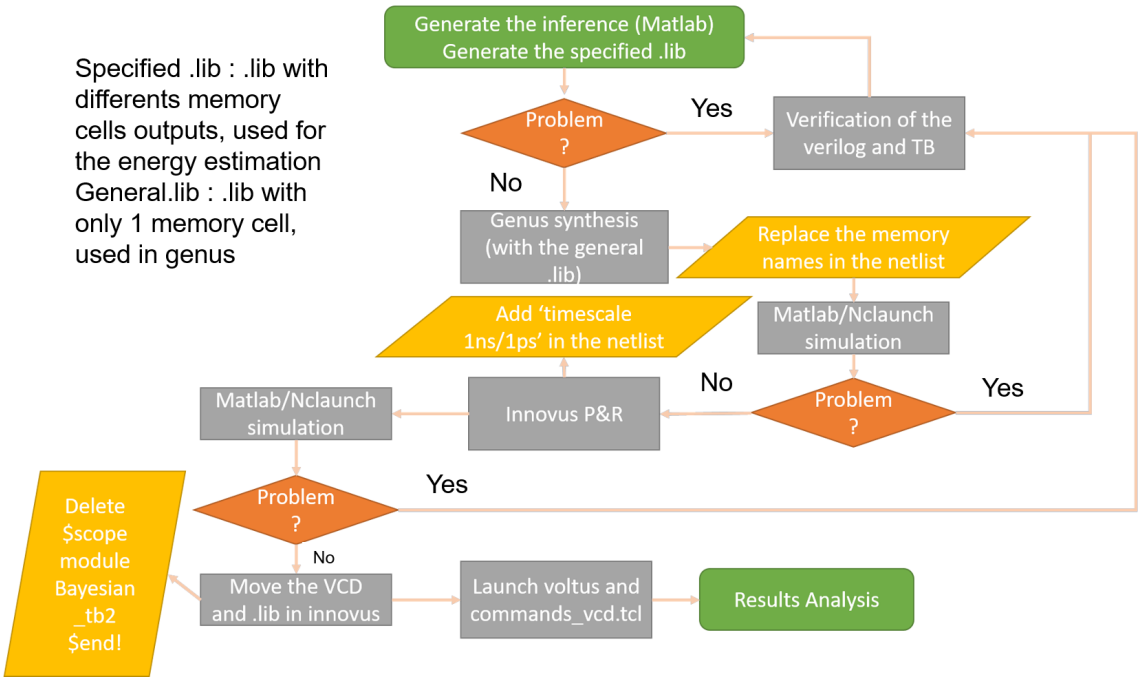


Figure 3.11: **Étapes de la simulation de consommation énergétique des puces constituées de memristors.**

Résultats Nous avons conçu et mis en place ce système dans notre procédé de référence (processus de fonderie à faible consommation de 130 nanomètres) et avons évalué sa consommation d'énergie. Nous pouvons voir dans l'image des masques montrés à la Fig. 3.12b que dans cette conception mise à l'échelle, la zone des tableaux de memristors est maintenant dominante, par rapport à la périphérie de la mémoire et au câblage de la machine bayésienne.

La Fig. 3.12c montre la consommation d'énergie des différents éléments du système dans les trois phases d'opération (après que les probabilités aient été programmées). L'initialisation du LFSR consiste à charger les seeds des six LFSRs du circuit. Cette opération consomme 0,38 nanojoules ; elle doit être effectuée une fois lorsque le système est allumé, et n'a pas besoin d'être répétée tant que l'alimentation reste allumée. Elle reste donc une contribution mineure à la consommation d'énergie. Cette énergie pourrait également être réduite de manière substantielle en câblant en dur la valeur des seeds optimales (alors que les seeds sont chargées à partir d'entrées externes dans notre conception).

En revanche, l'opération de lecture de la mémoire doit être effectuée chaque fois qu'une nouvelle entrée est présentée au système, et consomme un total de 0,3 nanojoules, incluant à la fois l'énergie associée aux circuits de mémoire eux-mêmes et à la logique de contrôle numérique. L'inférence stochastique elle même, correspondant au calcul stochastique, consomme 2,2 nanojoules (en supposant que tous les 255 cycles des LFSR ont été effectués). Par

conséquent, en net contraste avec les architectures de type von Neumann[54], la consommation d'énergie du calcul est dominante par rapport à l'énergie pour accéder aux données, mettant en évidence les avantages du calcul proche de la mémoire.

Pour analyser plus en détail l'utilisation de l'énergie de notre architecture, la Fig. 3.12d détaille les différentes sources de consommation d'énergie pendant le calcul stochastique. La distribution de l'horloge représente 11% de la consommation d'énergie. Ce chiffre reste relativement modeste, car l'horloge n'est pas distribuée au sein de la machine bayésienne elle-même, mais seulement à la logique de contrôle numérique externe. La multiplication des probabilités elle-même (portes ET) et le transfert de la sortie des blocs vers le bloc suivant, par des fils horizontaux, ne représentent que 1% de la consommation d'énergie totale. La génération de nombres aléatoires (LFSR, circuits Gupta) représente 60% de la consommation d'énergie, et leur distribution par des fils verticaux 28%. Les efforts futurs devraient donc se concentrer sur cette partie.

Avant cela, une technique évidente pour réduire la consommation d'énergie est de réduire le nombre de cycles calculés pendant l'inférence stochastique. Cette réduction impacte naturellement la précision, comme le montre la Fig. 3.12e. Cette figure montre, pour la tâche de reconnaissance de gestes, la précision de la machine bayésienne, en fonction du nombre de cycles considérés. Les nombres supérieurs à 255 ne servent à rien, car 255 est la périodicité des LFSR de huit bits. Nous considérons la stratégie traditionnelle de calcul stochastique, et la stratégie qui va réduire la consommation énergétique : power-conscious. Pour rappel, dans l'approche traditionnelle, le système est exploité pendant un nombre fixe de cycles, et le geste reconnu est choisi comme la sortie qui a généré le plus grand nombre de uns. Dans la stratégie simplifiée réduisant la consommation énergétique, le calcul est arrêté dès qu'une des sorties du circuit produit un 'un', et cette sortie donne le geste reconnu. Nous voyons que dans les deux cas, des nombres de cycles aussi bas que 50 permettent d'approcher la précision obtenue avec 255 cycles. Sur la base de ces résultats, la Fig. 3.12f montre l'interaction entre la précision et la consommation d'énergie, en utilisant les deux stratégies. L'approche power-conscious consomme moins d'énergie que l'approche conventionnelle à précision équivalente. Cependant, la première est limitée à une précision de 86%, tandis que la seconde peut atteindre 89%. Dans l'ensemble, la réduction du nombre de cycles semble être une stratégie très efficace : dans l'approche conventionnelle, accepter une réduction de la précision de seulement un point de pourcentage permet de réduire la consommation d'énergie par un facteur de 2,9.

Pour évaluer l'efficacité énergétique de notre approche, nous avons également mis en œuvre la tâche de reconnaissance de gestes bayésiens sur une unité de microcontrôleur STM32F746ZGT6 de ST Microelectronics (MCU, intégrée sur une carte de test Nucleo-F746ZG). Les MCU sont de petits ordinateurs intégrant toute leur logique, leur mémoire volatile et non volatile sur une seule puce. Ils sont actuellement l'approche dominante pour fournir l'IA à la périphérie dans des contextes à contrainte énergétique[101]. Notre mise en œuvre a été programmée en langage C en utilisant l'environnement de développement intégré STM32 Cube de ST Microelec-

tronics et a été optimisée pour fonctionner sur le MCU (le calcul stochastique de notre puce de test est remplacé par une addition entière standard, et les probabilités ont été remplacées par des log-probabilités pour éviter les multiplications). Pour effectuer le benchmark, le MCU calcule la reconnaissance de gestes pour toutes les entrées possibles de manière séquentielle et fait clignoter une LED sur la carte à chaque million d'inférences pour permettre un chronométrage précis. La consommation d'énergie du MCU a été mesurée à l'aide d'un ampèremètre standard (la consommation d'énergie du seul MCU a été mesurée, excluant tous les autres composants de la carte). Pour isoler la consommation d'énergie strictement due à l'inférence bayésienne, nous avons également mesuré la consommation d'énergie d'un programme de contrôle qui comprend toutes les opérations effectuées par le programme de reconnaissance de gestes (bouclage des entrées, clignotement de la LED...), sauf l'inférence bayésienne réelle. Ensuite, nous soustrayons la consommation d'énergie du programme complet de reconnaissance de gestes ($2.4\mu J$/geste) et celle du programme de contrôle ($0.4\mu J$/geste) pour obtenir l'énergie utilisée par le MCU strictement pour l'inférence bayésienne ($2.0\mu J$/geste). En comparaison, la machine bayésienne, même lorsqu'elle utilise 255 cycles, utilise un total de 2,5 nanojoules pour reconnaître un chiffre en utilisant l'approche conventionnelle, et 0,4 nanojoules en utilisant l'approche power-conscious (avec la tension d'alimentation maximale de 1,2 volts). C'est particulièrement impressionnant car notre MCU de référence est fabriqué dans un nœud CMOS de 90 nanomètres, comparable mais plus économe en énergie que le nœud CMOS de 130 nanomètres utilisé pour notre machine bayésienne.

3.3.2 Conclusion et Perspectives d'Améliorations

Nous avons montré qu'une machine bayésienne peut être mise en œuvre dans un système avec des memristors distribués, effectuant des calculs localement et avec un minimum de mouvement d'énergie. Cela lui permet d'effectuer une inférence bayésienne avec une efficacité énergétique des ordres de grandeur supérieure à celle d'une unité de microcontrôleur standard. En raison de sa dépendance à la mémoire non volatile, et de son utilisation exclusive des opérations de lecture, une fois que les probabilités ont été programmées, le système peut être mis hors tension tout en retrouvant instantanément sa fonctionnalité. Il peut également être exploité à des tensions d'alimentation faibles, et éventuellement variables. Bien que les modèles bayésiens soient généralement considérés comme coûteux en termes de calculs, nos résultats suggèrent que des modèles complexes pourraient être intégrés dans des systèmes d'Internet des objets (IoT) par exemple, avec une faible consommation d'énergie. Cela pourrait permettre à ces systèmes de bénéficier des qualités de l'inférence bayésienne pour faire face à des situations très incertaines avec peu de données, et pour faire des prédictions en utilisant un mode explicable.

3.3.2.1 Comparaison avec les Réseaux de Neurones

Contrairement à ce que j'ai expliqué dans les sections 1.1.2 et 1.2.3 sur les résaeux de neurones, qui se basent sur des memristors utilisant le calcul analogique, au moins dans une certaine mesure, notre machine bayésienne est un design numérique. Les choix de conception de la machine bayésienne ont été guidés par les spécificités de l'inférence bayésienne. Le besoin de probabilités de huit bits, une précision supérieure à ce que les memristors analogiques peuvent fournir, et le fait que la loi de Bayes n'exige pas de multiplication et d'accumulation, ont conduit à une conception numérique. Cela nous a permis de nous appuyer sur un amplificateur de détection simple pour lire les memristors et non l'utilisation de convertisseurs analogique-numérique, ce qui apporte de multiples avantages :

- il est très flexible en termes de tension d'alimentation
- il fonctionne sans avoir besoin d'aucun étalonnage
- il atténue les perturbations de lecture
- il est largement insensible à la variation des dispositifs

La simplicité de l'amplificateur de détection nous a également permis de démontrer un système complet comportant 16 petits blocs de mémoire, alors que les réseaux neuronaux basés sur des memristors analogiques ont généralement un seul bloc de mémoire.

3.3.2.2 Perspectives d'Amélioration

Nos résultats ont été obtenus en utilisant un processus de 130 nanomètres, illustrant que l'efficacité énergétique peut être obtenue avec une technologie peu coûteuse. Comme la consommation d'énergie est dominée par la circuiterie numérique, elle pourrait également être réduite en adaptant la conception à des nœuds technologiques plus agressifs. Notre analyse énergétique a révélé que 88% de la consommation d'énergie pendant la phase d'inférence était due à la génération et à la distribution de nombres aléatoires. Le coût de génération est dû à l'utilisation de LFSR, et le coût de distribution est dû à la nature non locale de la génération de nombres aléatoires (notre système utilise un seul LFSR par colonne, partagé par tous les blocs de probabilités de la colonne).

L'efficacité énergétique de la machine bayésienne stochastique pourrait s'améliorer considérablement à l'avenir. Plusieurs travaux récents ont montré la possibilité de générer des bits aléatoires à un coût énergétique très faible en utilisant des nanodispositifs stochastiques tels que les jonctions tunnel superparamagnétiques[102–105] ou le bruit télégraphique aléatoire dans les RRAM[106]. Ces propositions reposent sur des fluctuations naturelles des dispositifs dues au bruit thermique ou au bruit télégraphique aléatoire et peuvent donc générer des bits aléatoires en se basant uniquement sur l'opération de lecture, à un coût très faible. Ils pourraient être distribués au sein des tableaux de probabilités, grâce à leur petite surface, et générer

des bits aléatoires à un coût énergétique beaucoup plus faible que les LFSR (de l'ordre du femtojoule/bit) et sans nécessiter de fils verticaux. Dans le cas particulier des 'p-bits', le circuit Gupta, qui consomme dans notre conception 22% de l'énergie d'inférence, pourrait également être évité, car ce concept fournit des bits aléatoires avec des probabilités facilement ajustables [104, 105].

Notons que d'autres travaux ont proposé de générer des bits aléatoires en exploitant les opérations d'écriture de mémoires émergentes (par exemple, dans les MRAM[107] ou dans les RRAM[108]). Ces propositions peuvent générer des bits aléatoires de haute qualité à grande vitesse, mais ne seraient pas nécessairement adaptées pour la machine bayésienne. En raison de la nécessité d'une opération d'écriture pour générer un bit aléatoire, leur consommation d'énergie est comparable ou supérieure à celle des LFSR utilisés dans la version actuelle de la machine.

Enfin, nous évaluons le choix du calcul stochastique pour effectuer l'inférence bayésienne. Un avantage majeur du calcul stochastique est sa tolérance intrinsèque aux erreurs uniques. Elle présente également des propriétés favorables en termes de consommation d'énergie. Nous avons démontré qu'une machine bayésienne peut être mise en œuvre dans un système avec des memristors distribués, effectuant des calculs localement et avec un minimum de mouvement, permettant une efficacité énergétique bien supérieure à celle d'une unité de microcontrôleur standard. Cette capacité à effectuer des inférences bayésiennes de manière efficace énergétiquement ouvre la voie à l'intégration de modèles complexes dans des systèmes embarqués (edge systems). Cependant, l'utilisation du calcul stochastique et plus particulièrement la génération des nombres aléatoires consomme énormément d'énergie. On propose ainsi dans le chapitre suivant une nouvelle méthode de calcul basée sur l'utilisation d'additionneurs pour effectuer la multiplication en un cycle d'horloge.

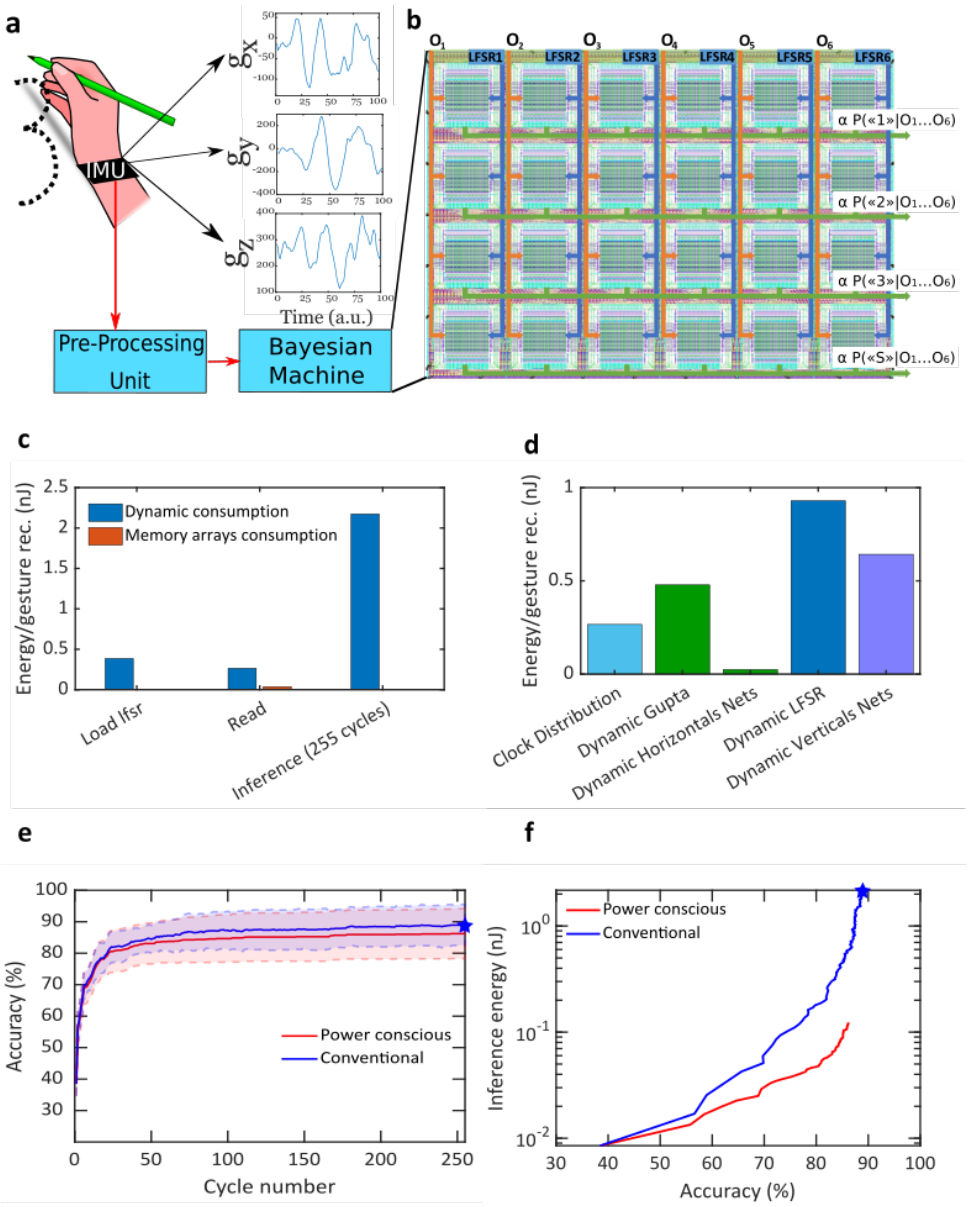


Figure 3.12: **Application de la machine bayésienne à une tâche pratique de reconnaissance de gestes. a** Configuration avec une unité de mesure inertielle utilisée pour enregistrer l'ensemble de données de reconnaissance de gestes. **b** Masques de la conception de la machine bayésienne placée et routée utilisée pour effectuer l'analyse de la reconnaissance de gestes au niveau de la conception. **c** Consommation d'énergie du système (consommation dynamique et matrices de mémoire) au cours des trois phases de calcul : chargement des *seeds* dans le LFSR, lecture des mémoires et inférence réelle de 255 cycles. **d** Consommation d'énergie des points importants du système pendant la phase d'inférence pour 255 cycles. **e** *Précision moyenne* en fonction du nombre de cycles pour deux types de calcul : en utilisant une méthode économe en énergie en ne prenant en compte que le premier '1' pour la décision (en rouge) et en utilisant le calcul stochastique conventionnel en utilisant le nombre maximum de '1' pour la décision (en bleu). Les ombres autour du graphique montrent un écart-type de la précision moyenne sur les dix sujets. **f** Consommation d'énergie pendant la phase d'inférence en fonction de la précision pour la reconnaissance de gestes pour les deux méthodes. Les étoiles correspondent au même point dans les graphiques **e** et **f**. Toutes les valeurs d'énergie sont données pour une tension d'alimentation de 1,2 volts.

Chapter 4

Machine Bayésienne III, les deux modes!

Une autre méthode de multiplier : le calcul logarithmique

> Un problème sans solution est un problème mal posé.
>
> Albert Einstein

LA MACHINE BAYÉSIENNE, malgré toutes ses qualités, n'est pas exemptée des défis du calcul stochastique. Ceux-ci incluent une latence élevée et une précision compromise lors de la manipulation de valeurs de probabilité faibles [89, 109]. Bien que la machine ait démontré une grande précision dans une application de reconnaissance gestuelle, son applicabilité plus large à diverses tâches demeure une question ouverte.

Dans ce chapitre, je vais explorer une approche alternative, passant des produits aux sommes. Toutes les valeurs de probabilité sont représentées logarithmiquement à l'aide d'entiers. Je définis dans un premier temps le calcul logarithmique et la nouvelle application utilisée, la classification des cycles du sommeil, qui est comparée à la reconnaissance de gestes. Dans un deuxième temps, j'étudie les mesures de la nouvelle puce sur la tâche des cycles du sommeil et également son intégration dans un coeur RISC-V. Dans un dernier temps, je discute d'une version agrandie de la machine bayésienne comprenant 131072 memristors et pouvant fonctionner dans les deux modes de calcul.

4.1 Une Machine Bayésienne Logarithmique

Premièrement, j'introduis un design alternatif : la machine bayésienne logarithmique basée sur des memristors. Ce design évite le calcul stochastique au profit du calcul logarithmique [109]. Cette modification se traduit par la mise en œuvre, au lieu de la multiplication des probabilités avec des multiplieurs tant coûteux sur le plan énergétique que sur l'encombrement, à une addition de celles-ci à l'aide d'additionneurs entiers numériques toujours proches de la mémoire. Je présente une machine bayésienne logarithmique entièrement fabriquée, utilisant des memristors à base d'oxyde d'hafnium/CMOS hybride présentés dans les chapitres précédents et démontre sa robustesse expérimentalement. Deuxièmement, j'entreprends une évaluation complète de cette machine, non seulement pour la reconnaissance gestuelle, mais aussi pour une application d'un filtre bayésien qui aborde une tâche dépendante du temps : la classification des phases du sommeil tout au long de la nuit. Une comparaison détaillée avec le design stochastique, évaluant la précision et la consommation d'énergie, clarifie davantage les scénarios dans lesquels chaque design excelle.

4.1.1 Le Calcul Logarithmique

Le calcul logarithmique est une méthode mathématique qui, dans le contexte de notre recherche, sert à simplifier la manipulation et la représentation des valeurs de probabilité. Cette approche est particulièrement utile pour traiter des valeurs de probabilité très faibles, qui sont difficiles à gérer avec précision dans le cadre du calcul stochastique. Le calcul logarithmique repose sur l'utilisation d'une fonction (eq. 4.1) pour transformer les produits en sommes, ce qui simplifie considérablement les opérations mathématiques. Ainsi, les valeurs de probabilité sont représentées logarithmiquement à l'aide d'entiers, permettant une représentation et

une manipulation plus efficaces et précises des valeurs de probabilité, même lorsqu'elles sont extrêmement faibles :

$$p = B^{\frac{n}{m}}. \tag{4.1}$$

Où les paramètres sont les suivants :

- n représente la valeur entière stockée dans la mémoire, associée à la probabilité p. Dans la suite, les entiers sont codés sur 8 bits.
- m représente une valeur modifiable qui va changer la forme de la courbe de la fonction comme le montre la figure 4.1, plus cette valeur est petite et plus la précision des faibles probilités est grande. Cette valeur est choisie comme étant égale à 8.
- B est la valeur pour laquelle la probabilité est égale à cette valeur quand $n = m$. Elle est choisie dans toute la suite à la valeur $B = 1/2$.

Les valeurs telles qu'elles sont codées impliquent que les valeurs entières de 0 à 8 correspondent à des probabilités allant de 1 à 1/2, tandis que les valeurs de 9 à 255 représentent des probabilités inférieures à 1/2. La plus petite probabilité qui peut être codée est $\frac{1}{2}^{255/8} \approx 2.5 \times 10^{-10}$. Par conséquent, ce codage privilégie les valeurs de probabilité inférieures, contrastant avec la méthodologie du calcul stochastique, où de telles valeurs sont difficiles à traiter.

La machine bayésienne logarithmique, comme illustré dans la figure 4.2a et b, conserve le principe architectural de son homologue stochastique. Les tableaux de memristors hébergent les vraisemblances $p(O_i|Y = y)$ sous forme d'entiers de 8 bits. Les observations d'entrée O_i fournissent des adresses de lignes pour ces tableaux, indiquant la valeur de vraisemblance spécifique à accéder. Ces valeurs sont ensuite agrégées à l'aide d'additionneurs entiers de 8 bits proches de la mémoire, qui se substituent aux portes ET de 1 bit trouvées dans le modèle stochastique. Les additionneurs sont conçus pour produire une sortie de 255 (c'est-à-dire, la probabilité minimale possible) dans les scénarios de débordement.

4.1.2 Une Nouvelle Application : les Cycles du Sommeil

Nous avons utilisé la détection des phases du sommeil tout au long de la nuit comme tâche de référence pour évaluer notre circuit intégré. Cette tâche a été sélectionnée pour sa représentativité dans des scénarios de calculs sur des systèmes embarqués (on-edge computing), où l'approvisionnement en énergie est limité. La classification des phases du sommeil peut fournir des informations précieuses sur les troubles du sommeil et la qualité globale du sommeil. Mathématiquement, cette tâche est plus complexe que la tâche de reconnaissance gestuelle utilisée pour évaluer la machine bayésienne stochastique. Une nuit de sommeil typique comprend différentes phases : éveil, Mouvement Rapide des Yeux (REM), sommeil léger et sommeil profond. Pour simplifier l'étude, nous avons regroupé les phases du sommeil léger et profond,

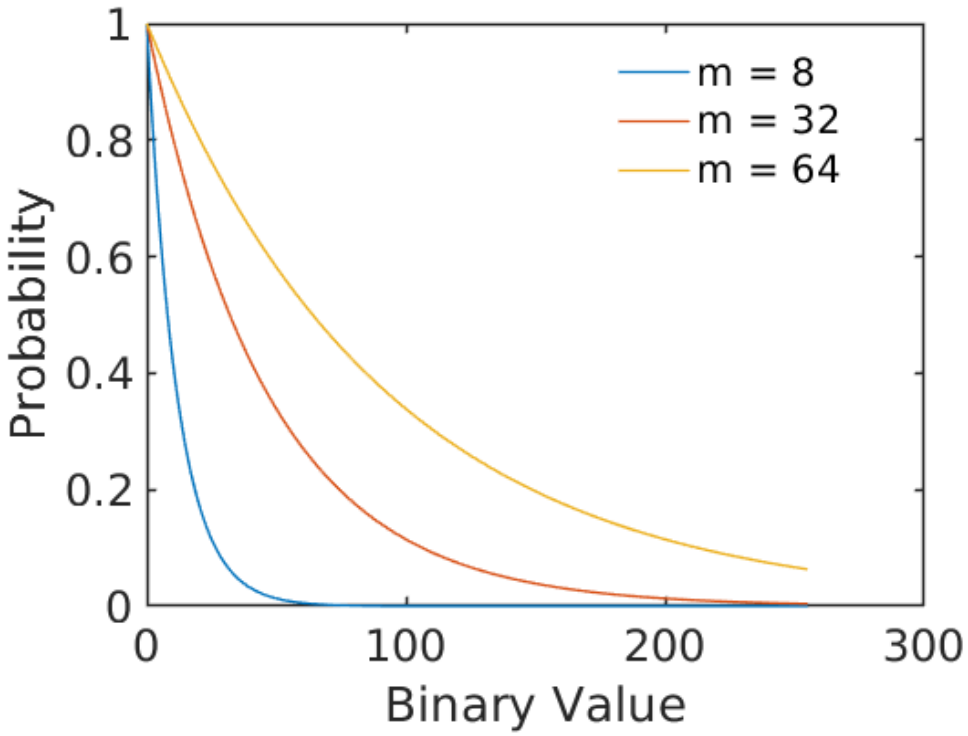


Figure 4.1: **Fonction logarithmique transformant la probabilité en entier 8 bits**, pour trois valeurs de m : 8, 32, 64.

qui sont parfois chacune séparées en deux phases distinctes. Notre objectif est de classer les phases du sommeil tout au long de la nuit en utilisant des mesures d'électroencéphalographie (EEG) et d'électromyographie (EMG).

Inférer les phases du sommeil uniquement à partir des lectures EEG et EMG est un défi. Le contexte temporel joue un rôle significatif : si un patient est identifié comme étant en sommeil profond au temps t, la probabilité de la même phase cinq secondes plus tard (notée $t+1$) est substantielle. Cette corrélation séquentielle est capturée à l'aide d'un modèle bayésien, comme représenté par la relation suivante, obtenue en utilisant la loi de Bayes :

$$p(Y(t+1) = y|EEGdata, EMGdata, Y(t)) \propto$$
$$p(EEGdata, EMGdata|Y(t+1) = y), Y) \times p(Y(t+1) = y|Y(t)). \quad (4.2)$$

En partant du principe que les données EEG et EMG sont principalement influencées par l'état de sommeil au temps t, et que cette influence reste constante, nous raffinons l'équation

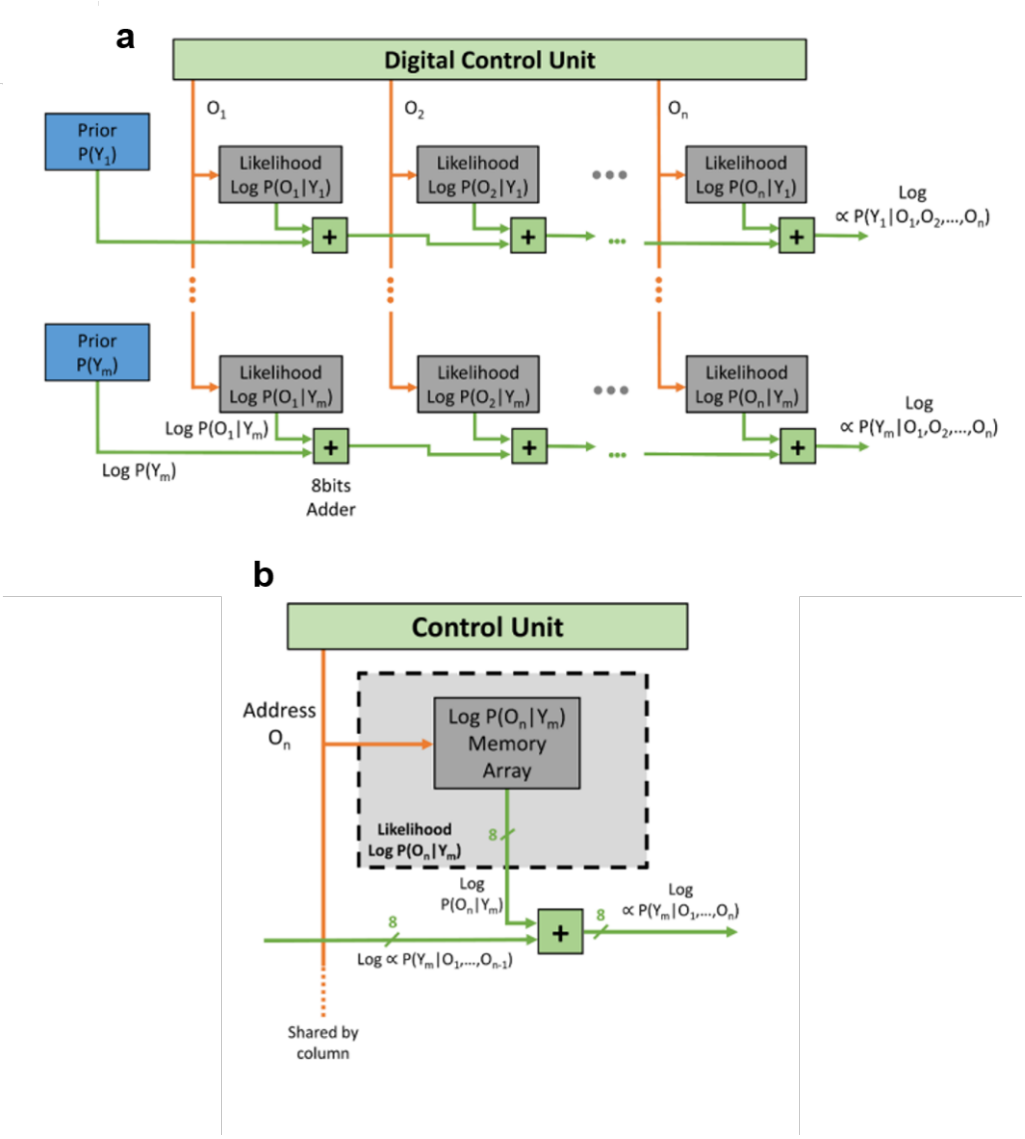


Figure 4.2: **Architecture générale de la machine bayésienne logarithmique. a** Schéma simplifié de la machine bayésienne logarithmique mise en œuvre. **b** Schéma simplifié de la likelihood comprenant les additionneurs.

comme :

$$p(Y(t+1) = y|Y(t), EEGdata, EMGdata) \propto$$
$$p(Y(t+1) = y|Y(t)) \times p(EEGdata, EMGdata|Y = y)). \quad (4.3)$$

Nous avons obtenu les données des patients à partir de l'ensemble de données DREAMS composé de 20 enregistrements de polysomnographie (PSG) annotés avec les phases du sommeil de toute une nuit provenant de sujets sains [110]. Nous n'avons pas utilisé les données d'électrooculographie disponibles dans celui-ci. Il est important de souligner que notre mise

Phase du Sommeil	EEG	EMG
Éveillé	Ondes Alpha (8-13 Hz)	Tonus élevé
Sommeil REM	-	Tonus faible
Sommeil non-REM léger	Ondes alpha (8-13 Hz) et theta (4-8H z)	-
Sommeil non-REM profond	Ondes delta (0,5-4 Hz)	-

Table 4.1: Résumé des caractéristiques distinctives des phases du sommeil en termes de présence d'ondes spécifiques dans l'EEG et de niveau de tonus dans l'EMG.

en œuvre offre une étude de cas de preuve de concept pour notre machine bayésienne, qui, bien qu'efficace, n'optimise pas la précision potentielle qu'on pourrait obtenir en utilisant l'ensemble des données de manière plus approfondie.

Dans l'ensemble de données, les phases du sommeil ont été annotées selon les critères de Rechtschaffen et Kales. Nous avons simplifié ces données en quatre catégories pour s'adapter à la capacité de quatre sorties de la puce : état éveillé, sommeil REM, sommeil non-REM léger (combinant les phases 1 et 2 de l'Académie Américaine de Médecine du Sommeil, ou AASM), sommeil non-REM profond (phase 3 de l'AASM).

Sur la base de l'analyse du tableau 4.1, et de l'optimisation de la précision de l'ensemble d'entraînement, nous avons choisi trois observables. Chaque échantillon de cinq secondes a subi une Transformée de Fourier Rapide (FFT) pour l'analyse du signal EEG. La densité spectrale de puissance aux fréquences de 1,5Hz et 9,35Hz a été désignée comme les deux premières observables. La puissance totale du signal EMG a également été calculée et utilisée comme troisième observable. Ensuite, les vraisemblances des modèles bayésiens pour chaque phase ont été estimées à partir des données d'entraînement, et modélisées en utilisant une distribution log-normale. Les probabilités a priori $p(Y(t+1)|Y(t)$ ont été ajustées en fonction des phases du sommeil précédentes pour améliorer la performance du modèle.

En supposant l'indépendance conditionnelle entre nos trois observations, notre modèle bayésien raffiné est alors représenté comme :

$$p(Y(t+1) = y|Y(t), EEGdata, EMGdata) \propto$$
$$p(Y(t+1) = y|Y(t)) \times p(lowfreq.EEG|Y = y)) \times p(highfreq.EEG|Y = y)) \times p(EMG|Y = y)). \tag{4.4}$$

Implémenter cette équation dans notre machine bayésienne est intuitif, en utilisant la prédiction bayésienne précédente comme une entrée et les trois observations comme les trois autres entrées de la machine (voir fig. 4.3a). Plus formellement, ce modèle effectue une inférence sur le réseau bayésien présenté dans la figure 4.3b.

Pour l'entraînement et le test du modèle, nous avons sélectionné le premier sujet sain dans la base de données DREAMS. Un ensemble de 40 échantillons de cinq secondes pour chaque phase de sommeil a été utilisé pour l'entraînement, tandis que les données restantes ont servi

de jeu de données de test. Cet ensemble d'entraînement restreint a été intentionnellement choisi pour démontrer l'efficacité dans un contexte de faibles données, ce qui est une caractéristique distinctive des approches bayésiennes (voir section 2.3.4) [85, 98].

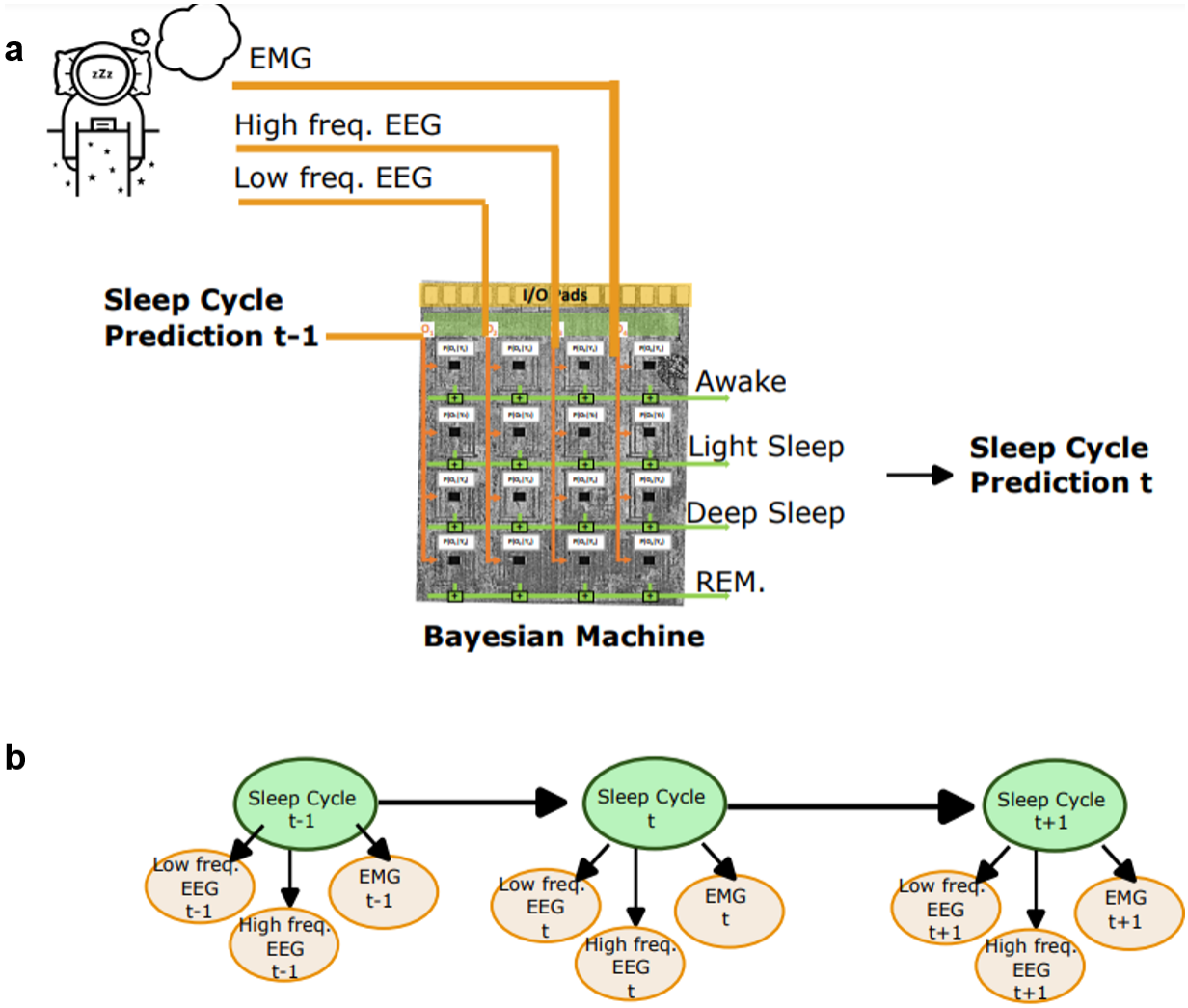


Figure 4.3: **Mise en œuvre expérimentale de la classification des phases du sommeil tout au long de la nuit. a** Entrées et sorties de la machine bayésienne logarithmique utilisée pour la classification des phases du sommeil. **b** Modèle de réseau bayésien implémenté sur la machine bayésienne logarithmique utilisée pour la classification des phases du sommeil.

4.1.3 Comparaison des Deux Méthodes de Calcul

Je me penche maintenant sur une analyse comparative des machines bayésiennes stochastiques et logarithmiques, en utilisant deux tâches clés comme références : la reconnaissance des gestes, précédemment discutée dans la section 2.3, et la classification des phases du som-

meil tout au long de la nuit que je viens d'introduire.

4.1.3.1 Évaluation des Performances sur les Deux Applications

Gesture Recognition

Sleep Recognition

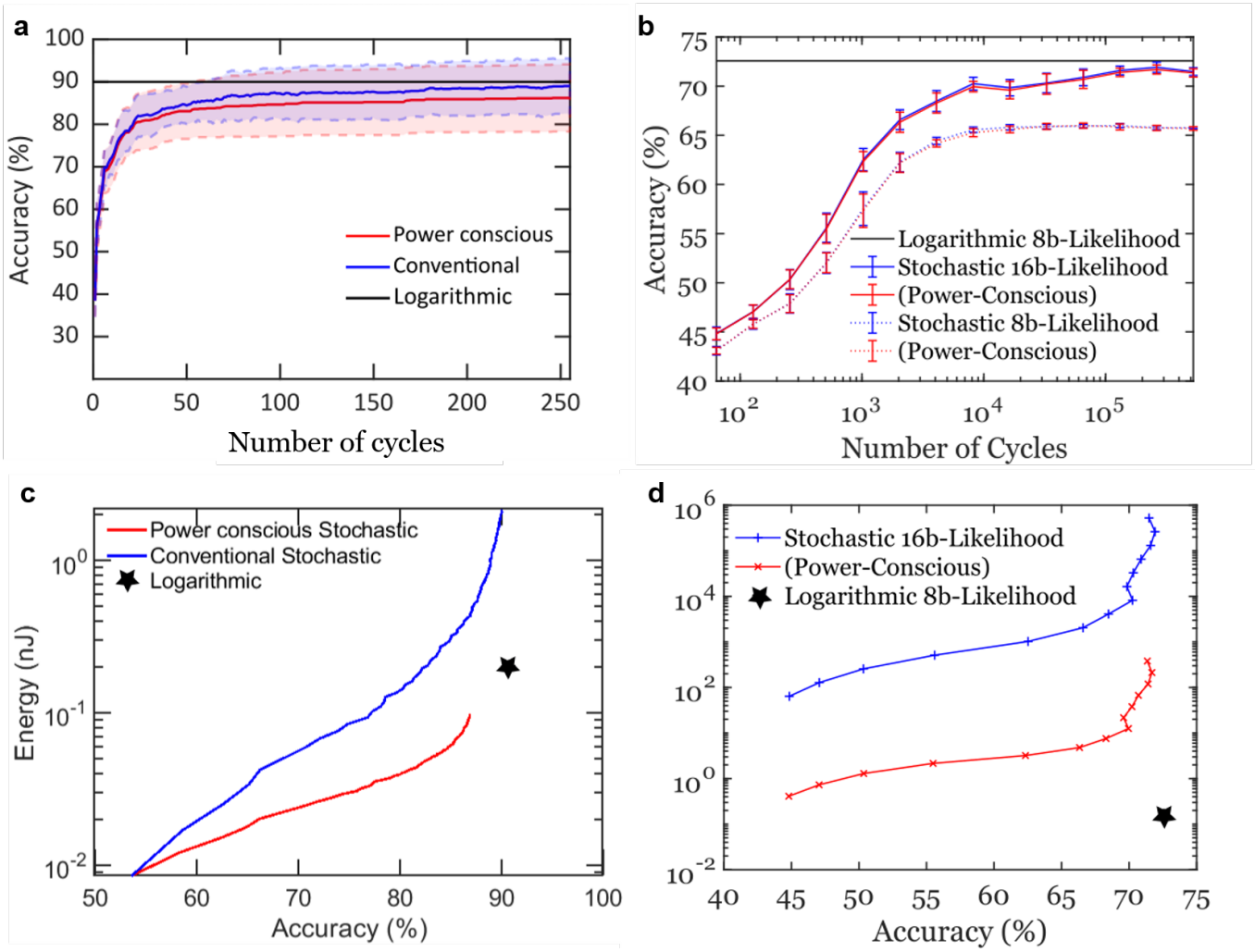


Figure 4.4: **Évaluation de la précision et de la consommation d'énergie des machines bayésiennes stochastiques et logarithmiques.** **a,b** Précision de la machine stochastique pour **a** la reconnaissance des gestes et **b** la classification des phases du sommeil, en fonction du nombre de cycles d'horloge, en utilisant le calcul stochastique conventionnel ou l'approche "power-conscious". La précision de la machine logarithmique (un cycle d'horloge) est tracée comme référence. **c,d** Consommation d'énergie en fonction de la précision pour **c** la reconnaissance des gestes et **d** la classification des phases du sommeil, en utilisant toutes les approches considérées dans a et b. Cette figure est obtenue en associant plusieurs méthodologies de simulation. Les barres d'erreur/ombres représentent un écart type.

Une caractéristique du calcul stochastique est sa sortie intrinsèquement moyennée sur plusieurs cycles d'horloge. Ainsi, une augmentation du nombre de cycles d'horloge améliore directement la précision des résultats. Cependant, cette augmentation de précision se fait au détriment d'une latence plus longue et d'une consommation d'énergie élevée. La figure 4.4a visualise la précision pour la reconnaissance des gestes par rapport au nombre de cycles d'horloge. La machine bayésienne logarithmique, équipée de vraisemblances logarithmiques de 8 bits, est établie comme un point de référence. Les résultats stochastiques présentés, sont comme dans la section 3.3.1, des résultats simulés d'une machine bayésienne agrandie par rapport à la version fabriquée, obtenus à l'aide de simulations précises en termes de cycles. J'affiche également les deux méthodes du calcul stochastique vues dans la section 2.3.1.1

À partir des données affichées dans la figure 4.4a, nous discernons que le calcul stochastique, même avec seulement 50 cycles, fournit des résultats de qualité : 86.7% de précision pour la stratégie conventionnelle et 84.4% pour l'approche "power-conscious". Élever le nombre de cycles à 255 booste ces chiffres à 89% et 86.9%, respectivement, tandis que la machine logarithmique atteint une précision de 90.6%.

En passant à la tâche de classification des phases du sommeil, les défis posés au calcul stochastique deviennent plus prononcés. Comme le démontre la figure 4.4b, les calculs stochastiques basés sur des vraisemblances de 8 bits atteignent une précision maximale de 65%, une réduction nette par rapport aux 73% atteints par l'approche logarithmique. Passer à des vraisemblances de 16 bits dans le calcul stochastique améliore sa précision à 72%, mais nécessite un nombre très élevé de cycles d'horloge. 10 000 cycles atteignent à peine 70% de précision. De manière remarquable, contrairement à la tâche de reconnaissance des gestes, l'approche "power-conscious" correspond étroitement à la performance du calcul stochastique conventionnel.

La raison derrière ces résultats est intuitive : la tâche des phases du sommeil traite intrinsèquement de faibles valeurs de vraisemblance, une conséquence de ses caractéristiques dépendantes du temps. Plus précisément, lorsqu'on est dans une phase de sommeil donnée, la probabilité de passer à une autre phase à l'étape de temps suivante, notée $p(Y(t+1) = y2|Y(t) = y1)$, est limitée. Cependant, des données EEG et EMG robustes peuvent contrer ces probabilités. Ce scénario exige intrinsèquement la gestion de faibles probabilités, un domaine où le calcul stochastique est en difficulté, tandis que le calcul logarithmique excelle. Par conséquent, la complexité de la gestion de faibles probabilités dans des scénarios complexes éclaire pourquoi le calcul stochastique basé sur des vraisemblances de 8 bits est insuffisant pour le calcul stochastique. De plus, cette complexité explique le grand nombre de cycles d'horloge requis par le calcul stochastique, ainsi que la performance comparable de l'approche "power-conscious" et du calcul stochastique conventionnel.

J'étends maintenant notre analyse comparative à la consommation d'énergie des machines. Pour quantifier cette métrique, j'utilise la méthodologie d'évaluation détaillée dans la section 3.3. Ces évaluations sont basées sur le processus CMOS de 130 nanomètres et la tech-

nologie des memristors à oxyde de hafnium utilisée dans notre puce de test.

La Figure 4.4c trace la consommation d'énergie par tâche de reconnaissance des gestes en fonction de la précision pour les deux stratégies du calcul stochastique - conventionnelle et "power-conscious" - et la machine logarithmique à vraisemblance de 8 bits. Le graphique révèle que l'approche "power-conscious" surpasse constamment la machine logarithmique en termes d'efficacité énergétique. Cependant, elle a un plafond de précision de 86%, n'atteignant pas la précision de 90% atteinte par la méthode logarithmique. D'autre part, le calcul stochastique conventionnel n'obtient un avantage énergétique que pour des précisions inférieures à 80%, le rendant non compétitif avec la machine logarithmique.

Les résultats relatifs à la tâche de classification des phases du sommeil, présentés dans la figure 4.4d, indiquent un paysage radicalement différent. Ici, l'approche logarithmique surclasse les deux stratégies stochastiques à tous les niveaux de précision. Bien que la méthode "power-conscious" montre des avantages significatifs en matière d'économie d'énergie par rapport à son homologue conventionnel, elle reste largement moins compétitive que la machine logarithmique. Ces découvertes accentuent davantage l'adaptabilité de la machine logarithmique aux tâches dépendantes du temps impliquant de petites valeurs de probabilité.

4.1.3.2 Résilience aux erreurs

Au-delà de la performance et de l'efficacité énergétique, la robustesse aux erreurs de lecture de mémoire est un défi important dans les machines bayésiennes basées sur des memristors. La figure 4.5 présente les résultats de simulations de Monte Carlo, intégrant des erreurs artificiellement induites dans les tableaux de memristors. À travers les deux tâches, toutes les stratégies montrent une résilience considérable aux taux d'erreur de bit de mémoire, une caractéristique attribuable à la nature même des tâches d'apprentissage automatique. De telles tâches sont généralement robustes aux erreurs dans les paramètres en raison des redondances inhérentes—par exemple, plusieurs observations peuvent fournir des informations redondantes.

Nous voyons dans la figure 4.5 que les approches stochastiques montrent une résilience accrue par rapport à l'approche logarithmique, mais pas pour les raisons que l'on pourrait initialement supposer. Puisque la mémoire est lue une fois par présentation d'entrée, une erreur n'est pas lissée par le calcul stochastique ; au contraire, elle impacte l'ensemble du calcul. La plus grande robustesse des machines stochastiques est plutôt attribuable à la représentation linéaire des vraisemblances, où les erreurs de bit-flip sont, en moyenne, moins conséquentes que dans des représentations logarithmiques.

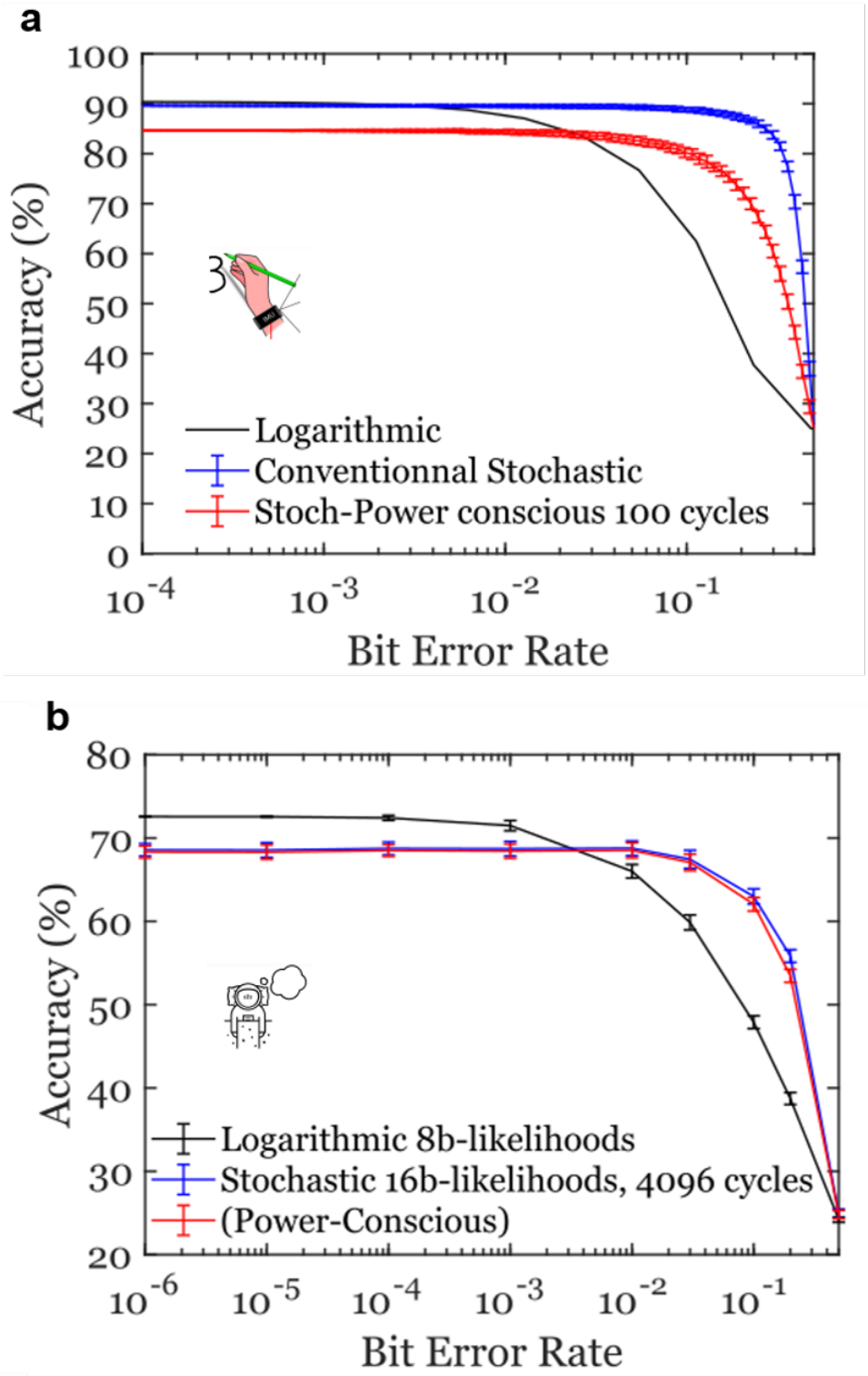


Figure 4.5: **Évaluation de la Résilience au taux d'erreur de bit de memristor de la machine bayésienne stochastique et logarithmique.** Précision des machines bayésiennes stochastiques (utilisant le calcul conventionnel et "power-conscious") et logarithmiques en fonction des taux d'erreur de bit de memristor, pour **a** la reconnaissance des gestes et **b** les tâches de classification des phases du sommeil. Cette figure est obtenue en utilisant la simulation de Monte Carlo. Les barres d'erreur/ombres représentent un écart type.

4.2 Mesures de la Nouvelle Puce

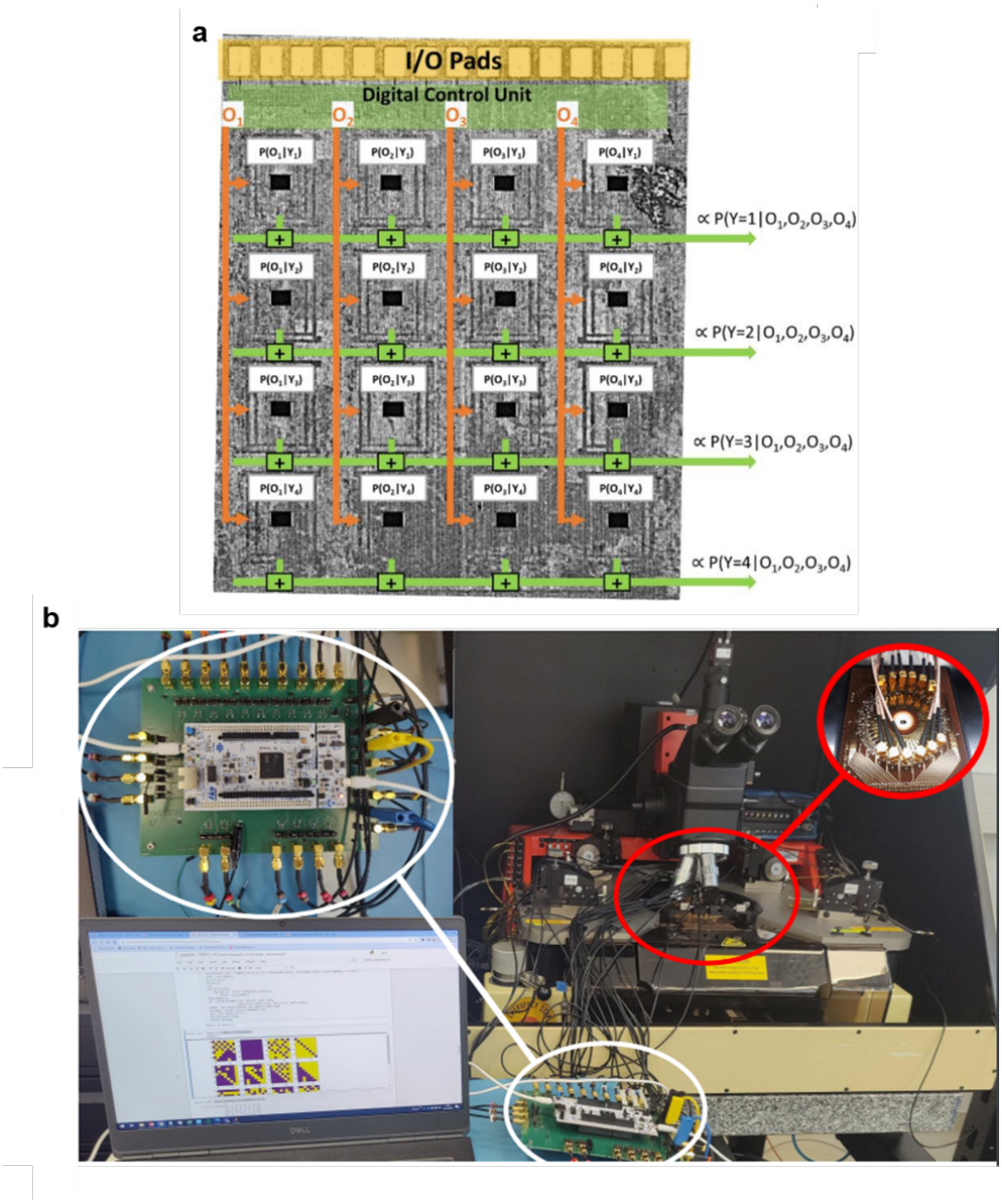


Figure 4.6: **Puce bayésienne logarithmique et son système de test.** **a** Image de microscopie optique de la puce de la machine fabriquée. **b** Photographie de la configuration du test de la machine bayésienne logarithmique.

4.2.1 Reconnaissance des cycles du sommeil

Comme dans le chapitre 3, la puce est fabriquée dans un processus 130 nm puis testée sous pointes comme le montre la figure 4.6b. La différence majeure dans ce système est le test des

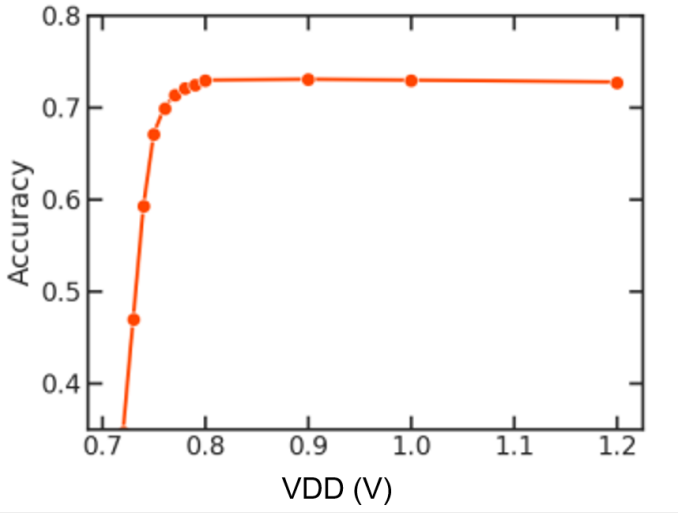


Figure 4.7: **Résulat de mesures de la puce bayésienne** Précision mesurée expérimentalement de la machine bayésienne logarithmique, moyennée sur 1000 segments de cinq secondes. La mesure a été répétée pour différentes tensions d'alimentation VDD.

inférences qui utilisent à présent le calcul logarithmique. Pour réaliser les tests, nous avons converti les vraissemblances dans leurs formes logarithmiques, et les avons ensuite programmées dans notre machine bayésienne pour validation expérimentale. Nous avons évalué la performance de notre machine en utilisant les 1000 dernières mesures d'une nuit de patient de l'ensemble de données DREAMS (qui n'ont pas été utilisées pour entraîner la machine). Le sous-ensemble de 1000 mesures a été choisi pour trouver un équilibre entre exhaustivité et faisabilité expérimentale. Les mesures de la figure 4.7 sont les résultats des inférences sur ces 1000 dernières données de la nuit et on peut remarquer que la puce fonctionne également bien à faible tension. On peut supposer que l'inflection que l'on peut observer dans cette figure est due à un problème de timing. En effet, quand la tension d'alimentation diminue, les systèmes numériques ralentissent et dans notre cas peuvent produire des erreurs. Une autre supposition possible est l'erreur de lecture, une diminution de la tension pourrait affecter les amplificateurs de détection qui peuvent alors lire une mauvaise valeur et comme nous avons vu, dans la figure 4.5, le calcul logarithmique est moins robuste aux erreurs que le calcul stochastique. Ce sont les raisons qui pourraient expliquer pourquoi la puce commence à perdre en précision à partir de 0.8 V alors que la puce stochastique fonctionnait jusqu'à 0.65 V.

4.2.2 Intégration de la Machine Bayésienne dans un Coeur RISC-V

Les machines bayésiennes sont compatibles avec des processeurs traditionnels comme nous avons pu le démontrer lors des tests précédents (voir section 3.2 et 4.2.1). Cependant, nous voulons pousser l'intégration des machines bayésiennes encore plus loin. Nous avons donc conçu un système complet avec un coeur RISC-V utilisant la machine bayésienne comme ac-

célérateur d'IA le schéma du système est montré en figure 4.8b. La figure 4.8a montre quant à elle la configuration du test avec la machine bayésienne logarithmique testée sous pointes et d'un coeur RISC-V prototypé sur un FPGA. Ce coeur (Ibex) est développé par lowRISC [111] et consiste en un processeur Harvard connecté au réseau sur puce par un bus de données et par un bus d'instruction à une mémoire dédiée comme le montre la figure 4.8b. Le réseau sur puce interconnecte le coeur, les périphériques, la mémoire ainsi que l'accélérateur (la machine bayésienne).

En ce qui concerne les résultats associés à cette intégration, nous avons utilisé ce nouveau setup pour étudier de l'impact des conditions de programmation des memristors sur la lecture de ceux-ci (Fig. 4.8c). Ce graphique représente la précision du calcul bayésien par les machines logarithmiques, basée sur des statistiques sur 4 096 entrées différentes, lorsque les mémoires résistives ont été programmées avec différents courants de compliance (c'est-à-dire, tensions de grille du transistor de sélection), et en utilisant différentes tensions d'alimentation VDD. Des courants de compliance plus élevés sont nécessaires pour une précision maximale ; néanmoins, la précision de la machine diminue seulement progressivement lors de la réduction du courant de compliance, montrant sa résistance naturelle à l'imperfection de la mémoire. Cette Figure met également en évidence la large gamme de tensions d'alimentation VDD sur laquelle la machine fonctionne, comme nous avons pu le remarquer sur les figures 4.7 et 3.8.

La réalisation de ces mesures a été effectuée en codant un programme C qui va implémenter les instructions nécessaires à la réalisation des mesures puis de le programmer dans le coeur qui va ensuite sans aide extérieure faire les mesures.

Grâce à cette intégration, la durée nécessaire pour réaliser une très grande quantité de mesures successives comme celles de la figure 4.8c a été très faible comparée à la durée d'une machine bayésienne non intégrée dans un coeur RISC-V. Cette méthode a démontré que l'intégration de ces composants dans un système sur puce offre de meilleures performances en métrologie, rendant le système plus facilement utilisable et programmable en C, avec une précision de mesure accrue. En effet, ce qui a été fait précédement (voir section 3.2), c'est programmer un microcontrôleur, puis le contrôler à l'aide d'instructions Python envoyées en série, or la communication série est très lente comparée à des instructions sur FPGA.

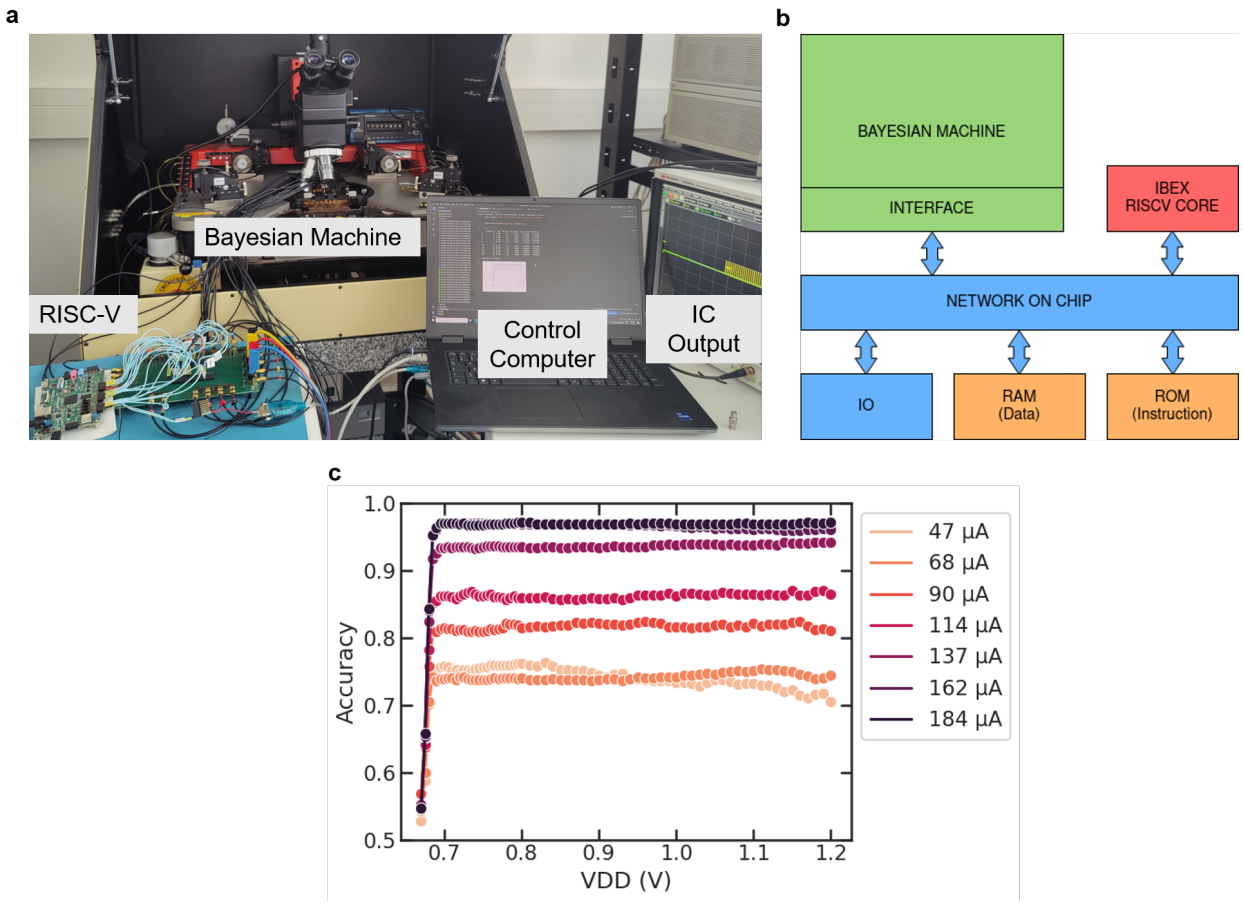


Figure 4.8: **Intégration des machines bayésiennes avec un cœur RISC-V. a** Configuration du test expérimental avec machine bayésienne et cœur RISC-V. **b** Schéma de l'intégration de la machine bayésienne dans un coeur RISC-V. **c** Précision mesurée de la sortie de la machine bayésienne logarithmique (moyenne sur 4 096 entrées), en fonction de la tension d'alimentation, pour différents courants de compliance SET (contrôlés par la tension de grille du transistor de sélection).

4.3 Vers une Version Agrandie et Multi-Modes de la Machine Bayésienne

Les études de notre machine bayésienne dans cette thèse ont montré des résultats prometteurs basés sur des mesures prises à partir de puces de démonstration et des estimations d'énergie sur des conceptions à grande échelle. Ces découvertes nous ont inspiré à pousser le projet plus loin et à fabriquer un système à grande échelle avec une mémoire plus grande et une capacité de calculs plus élevée qui peut gérer des tâches réelles sur la puce. Pour atteindre cet objectif, nous avons mis en œuvre une puce à grande échelle qui offre à la fois des modes de calculs logarithmique et stochastique. Le mode de calcul logarithmique offre une haute précision à un coût de calcul moindre, tandis que le mode de calcul stochastique est bien adapté pour des

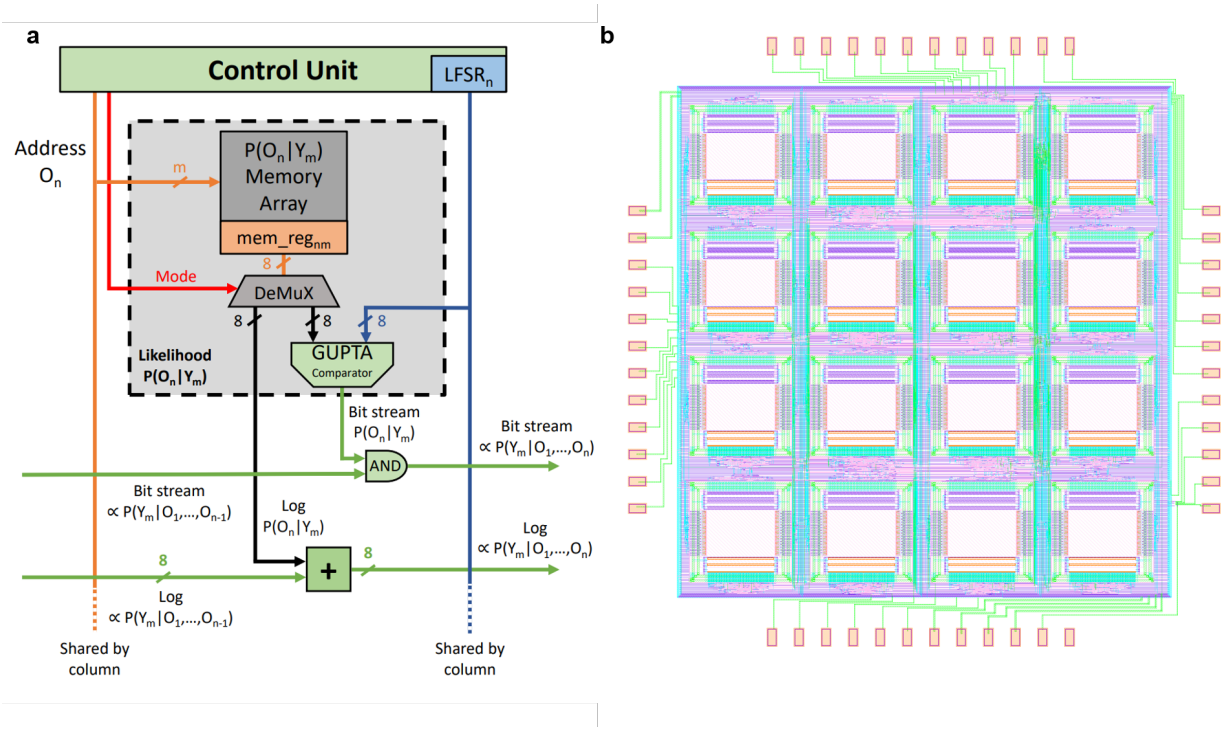


Figure 4.9: **Architecture de la puce Multi-mode.** **a** Architecture de la likelihood permettant l'utilisation des deux modes de calcul [112]. **b** Masques du design de la machine bayésienne à grande échelle placée et routée envoyés pour fabrication.

applications à faible consommation d'énergie, dans des environnements difficiles. En utilisant une machine bayésienne à multi-mode de calculs, nous pouvons profiter des avantages des deux modes de calculs en une seule puce. Pour rendre cela possible, nous avons utilisé un mécanisme qui permet à la machine bayésienne de basculer entre le mode du calcul logarithmique et le mode du calcul stochastique en fonction des spécifications de la tâche.

Dans cette section, je présente la version à grande échelle de la machine bayésienne, qui a été conçue et réalisée (voir Fig. 4.3b). Cette puce dispose de 4x4 blocs de vraisemblances avec une capacité mémoire de 64x64 bits (8k-memristor) et environ 143k memristors (131k pour la mémoire et 12k pour les factices) et 285k transistors. Cette capacité est suffisante pour tester des applications complexes et permettre des démonstrations pratiques telles que des capteurs intelligents, des caméras intelligentes ou des robots simples.

Bien que les deux modes partagent les mêmes tableaux de mémoire ainsi que les circuits de lecture et d'écriture, les circuits de calcul sont par contre séparés. La version stochastique utilise des circuits LFSRs et des vraisemblances avec un circuit comparateur et porte ET, tandis que la version logarithmique utilise la log-vraisemblance avec le circuit additionneur (voir 4.3a). Comme les probabilités sont stockées différemment dans chaque mode de calculs, le mode doit être déterminé avant la programmation de la puce.

Au moment où je termine l'écriture de ce manuscrit, la puce est en train d'être testée dans un processus utilisant le coeur RISC-V.

Conclusion Cette étude a exploré les performances et la robustesse des machines bayésiennes, particulièrement la machine bayésienne logarithmique qui utilise le calcul logarithmique, afin de surmonter les défis du calcul stochastique tels que la latence élevée et la précision sur certaines applications comportant de faibles probabilités. Ce design alternatif, basé sur des memristors et pouvant être intégré avec un cœur RISC-V, s'est révélé précis et économe en énergie en particulier dans des applications comme la reconnaissance gestuelle et la classification des phases du sommeil. Cependant, il est crucial de souligner que malgré leurs avantages, les machines bayésiennes logarithmiques ne surpassent pas toujours les designs stochastiques dans tous les scénarios, notamment en termes de résilience aux erreurs où les approches stochastiques ont montré une résilience accrue. Enfin, l'utilisation des nouvelles puces de plus grande capacité mémoire et leur intégration dans un système sur puce ouvrent la voie à de futures recherches.

Conclusions et Perspectives

> La presse, la machine, le chemin de fer, le télégraphe sont des prémisses dont personne n'a encore osé tirer la conclusion qui viendra dans mille ans.
>
> — Friedrich Nietzsche

Résumé et conclusions

Dans ce travail, nous avons exploré les défis et les opportunités associés à l'intégration de l'intelligence artificielle (IA) dans des environnements décentralisés, en mettant l'accent sur les applications mobiles et les systèmes embarqués. L'IA, qui a catalysé des avancées significatives dans divers domaines, opère principalement dans des environnements cloud, centralisant ainsi une quantité massive de données et créant des préoccupations en matière de sécurité des données. Nous avons abordé les défis de la consommation d'énergie et de la transparence, en développant des circuits intégrés spécialisés pour soutenir des modèles d'IA à faible consommation d'énergie et en incorporant l'inférence bayésienne pour améliorer la transparence et la fiabilité de l'IA.

Dans le chapitre 1, nous avons exploré l'innovation des transistors, des mémoires et des architectures innovantes s'inspirant du cerveau. Tout d'abord, ces innovations étaient à la recherche de meilleures performances en suivant la loi de Moore. Puis ces innovations sont passées à la recherche de l'optimisation de l'efficacité énergétique des circuits et l'évolution des architectures, notamment celle de Von-Neumann et des architectures neuromorphiques de calculs proches de la mémoire. Nous avons approfondi notre compréhension des memristors, une technologie de mémoire révolutionnaire non volatile, et de leur intégration pour le calcul en mémoire, jetant ainsi les bases pour des avancées significatives dans le domaine de l'informatique, de l'intelligence artificielle dans les systèmes embarqués.

Dans le chapitre 2, nous nous sommes intéressés à l'inférence bayésienne en étudiant les principes fondamentaux de l'approche bayésienne et une machine bayésienne qui est intégrée dans un circuit. Nous nous sommes penchés sur une méthode de calcul qui permet de diminuer la consommation énergétique et l'encombrement des multiplicateurs nécessaires à l'inférence bayésienne, le calcul stochastique. Nous avons exploré en détail l'application pratique de la reconnaissance des gestes, mettant en lumière l'importance de l'explicabilité et la performance sur des ensembles de données réduits. Cette exploration a révélé le potentiel des machines bayésiennes pour traiter des situations incertaines avec peu de données, tout en offrant des prédictions explicables.

Dans le chapitre 3, nous nous sommes consacrés à présenter la conception et la réalisation de la machine bayésienne sur une puce fabriquée dans un processus hybride CMOS/memristor. Nous avons mis en lumière les innovations et les défis associés à la mesure et à la caractérisation de ces puces. Les résultats obtenus ont démontré l'efficacité énergétique de la machine bayésienne, même en utilisant un processus de 130 nanomètres, et ont révélé que la consommation d'énergie pendant la phase d'inférence était principalement due à la génération et à la distribution de nombres aléatoires. Cette distribution peut encore évoluer en utilisant d'autres

nouvelles technologies comme les MRAM stochastiques. Cette efficacité énergétique ouvre la voie à l'intégration de modèles complexes dans des systèmes embarqués.

Dans le chapitre 4, nous avons exploré une approche alternative de machines bayésiennes, la machine bayésienne logarithmique, qui utilise le calcul logarithmique pour surmonter les défis du calcul stochastique. Cette approche s'est révélée précise et économe en énergie, en particulier dans des applications comme la reconnaissance gestuelle et la classification des phases du sommeil. Cependant, les machines bayésiennes logarithmiques ne surpassent pas toujours les designs stochastiques dans tous les scénarios. De plus, nous nous sommes penché sur une étude d'intégration de ces puces dans un système sur puce avec un cœur RISC-V qui ouvre la voie à de futures recherches et applications. Notamment avec la nouvelle puce, comprenant les deux modes de calculs et une plus grande capacité mémoire.

Perspectives

Mesures sur des puces plus performantes : La machine bayésienne à grande échelle, présentée dans la section 4.3, intégrant 131k memristors et deux modes de calcul, représente une avancée significative, offrant des perspectives prometteuses pour l'avenir de l'IA et d'autres applications nécessitant une grande capacité de stockage et de traitement des données. Cependant, l'exploration de technologies plus avancées et de nœuds technologiques plus avancés est cruciale. La transition vers ces nœuds pourrait non seulement réduire considérablement la consommation d'énergie, mais aussi améliorer les performances des circuits basés sur des memristors, permettant ainsi une efficacité énergétique supérieure, idéale pour l'informatique embarquée et les environnements à ressources limitées.
De plus, l'aspiration à développer un démonstrateur intégrant un processeur RISC-V, est une étape essentielle pour permettre la gestion de tâches plus complexes et l'avancement de la recherche dans les dispositifs intelligents et l'IA embarquée. L'intégration de différentes formes de mémoires résistives, comme les MRAM, pourrait également offrir des alternatives viables et efficaces, élargissant le spectre des applications possibles et optimisant davantage la consommation d'énergie et la résilience.
La finalité serait de développer une plateforme de recherche semblable aux micro-contrôleurs STM32 Nucléo ou aux FPGA DE2 d'Intel qui pourrait intégrer différents capteurs, micro-contrôleurs, FPGA et cellules solaires embarquées pour la collecte d'énergie (comme dans l'annexe B). Cette plateforme pourrait faire progresser la recherche dans les dispositifs intelligents et l'IA embarquée, servant des domaines d'application potentiels tels que l'informatique embarquée et les dispositifs de l'Internet des objets (IoT).

Association des réseaux de neurones et du raisonnement bayésien : Les réseaux de neurones et le raisonnement bayésien sont deux approches qui tentent de modéliser différents as-

pects du cerveau et de la cognition humaine, l'une mettant l'accent sur la structure et l'autre sur le processus de pensée. Comme je l'ai expliqué dans le chapitre 1, les réseaux de neurones artificiels tirent leur inspiration de l'architecture du cerveau humain. Ils sont composés d'unités, ou -neurones-, organisées en couches, qui sont connectées entre elles par des -synapses- pondérées. Cette structure permet aux réseaux de neurones de traiter l'information de manière parallèle et hiérarchique, similaire à la façon dont les neurones biologiques traitent l'information dans le cerveau. Les réseaux de neurones sont particulièrement efficaces pour apprendre des représentations de données complexes et non linéaires, ce qui les rend adaptés à des tâches telles que la reconnaissance d'images et le traitement du langage naturel.
Le raisonnement bayésien, en revanche, est une approche qui modélise la façon dont les êtres humains traitent l'incertitude et prennent des décisions en présence d'informations incomplètes ou ambiguës. Il est basé sur le théorème de Bayes, qui décrit comment mettre à jour les croyances (probabilités) en fonction de nouvelles données. Cette méthode reflète la capacité humaine à apprendre de l'expérience et à ajuster ses croyances en fonction de nouvelles informations, ce qui est un aspect crucial de la cognition humaine. Le raisonnement bayésien est souvent associé à une compréhension plus profonde et explicite des processus humains de pensée et de décision, offrant une interprétabilité et une justification claires des inférences réalisées.
Les deux approches offrent des perspectives uniques et complémentaires sur la compréhension et la réplication de l'intelligence humaine dans les systèmes artificiels, et pourraient être utilisées de manière complémentaire pour développer des modèles plus robustes, efficaces en énergie et explicables de l'intelligence artificielle.

Calcul en mémoire avec des memristors analogiques : Dans cette thèse, nous avons uniquement travaillé avec des memristors fonctionnant de manière numérique. Cependant comme je l'ai indiqué dans le chapitre 1, ils peuvent être utilisés de manière analogique. Nous avons fabriqué une puce permettant d'utiliser les memristors de manière numérique et analogique pour faire du calcul en mémoire, la description de cette puce est donnée dans l'annexe A. Elle peut être utilisée pour tester de nouvelles conceptions, affiner les concepts existants et approfondir notre compréhension des propriétés et des caractéristiques non idéales des memristors. De plus, elle crée une opportunité pour explorer les Réseaux Neuronaux Binaires et Ternaires basés sur des memristors, avec une fonction de plasticité synaptique inspirée du cerveau, qui pourrait être davantage optimisée avec une meilleure compréhension.

Apprentissage sur puce : La puce réalisée pour l'analyse des memristors (annexe A) peut de part l'utilisation des memristors en mode analogique être utilisée pour faire de l'apprentissage sur puce en utilisant les memristors. En effet, les memristors possèdent un régime de "Weak Reset" où la variation de résistance est linéaire en fonction des pulses appliqués [113]. Nous pouvons donc tester de nouveaux concepts et algorithmes neuromorphiques d'apprentissage,

qui sont compatibles avec le matériel ou adoptent des règles d'apprentissage locales pour la mise en œuvre et le test expérimental.
Une autre puce a également été designée pour réaliser l'algorithme d'échantillonnage de Monte-Carlo (MCMC) dont il est question dans l'annexe C. Cette puce intègre des circuits analogiques avancés, ouvrant la voie à la résolution des défis d'apprentissage in situ. Elle ouvre la possibilité d'intégrer des techniques bayésiennes plus sophistiquées, telles que les réseaux de neurones bayésiens. Cela pourrait potentiellement conduire à des améliorations substantielles tant en précision qu'en quantification de l'incertitude.

En conclusion, la recherche, tant du côté software, avec les différentes applications possibles de l'inférence bayésienne, que du côté hardware, avec l'intégration des nouvelles technologies mémoires comme les memristors et l'intégration des systèmes de collecte d'énergie, est une étape importante dans l'innovation de l'intelligence artificielle à faible consommation d'énergie et plus particulièrement de l'intelligence artificielle embarquée. Ces avancements permettent d'envisager des solutions plus durables et autonomes, capables de tirer parti de sources d'énergie renouvelables et d'optimiser la consommation énergétique, ce qui est crucial pour les applications en environnements contraints en ressources.

List of publications

Peer-Reviewed Journal Articles

✠ Kamel-Eddine Harabi[†], Tifenn Hirtzlin[†]**Clément Turck**[†], Elisa Vianello, Raphaël Laurent, Jacques Droulez, Pierre Bessière, Jean-Michel Portal, Marc Bocquet and Damien Querlioz, "A memristor-based Bayesian machine" , *Nature Electronics*, vol. 6, No. 1, p. 52–63, 2023. doi:10.1038/s41928-022-00886-9

✠ Thomas Dalgaty, Niccolo Castellani, **Clément Turck**, Kamel-Eddine Harabi, Damien Querlioz and Elisa Vianello, "In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling" , *Nature Electronics*, vol. 4, No. 2, p. 151–161, 2021. doi:10.1038/s41928-020-00523-3

✠ Fadi Jebali, Atreya Majumdar, **Clément Turck**, Kamel-Eddine Harabi, Mathieu-coumba Faye, Eloi Muhr, Jean-Pierre Walder, Oleksandr Bilousov, Amadeo Michaud, Elisa Vianello, Tifenn Hirtzlin, Francois Andrieu, Marc Bocquet, Stephane Collin, Damien Querlioz and Jean-Michel Portal, "Powering AI at the Edge: A Robust, Memristor-based Binarized Neural Network with Near-Memory Computing and Miniaturized Solar Cell" , *Under review at Nature Communications*.

Peer-Reviewed Conference Acts

✠ Harabi K.-E., **Turck C.**, Drouhin M., Renaudineau A., Bersani-Veroni T., Hirtzlin T., Vianello E., Bocquet M., Portal J.-M. and Querlioz D., "A Multimode Hybrid Memristor-CMOS Prototyping Platform Supporting Digital and Analog Projects" , *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, p. 184–185, 2023. doi:10.1145/3566097.3567944

✠ **Clément Turck**, Kamel-Eddine Harabi, Tifenn Hirtzlin, Elisa Vianello, Raphaël Laurent, Jacques Droulez, Pierre Bessière, Jean-Michel Portal, Marc Bocquet and Damien Querlioz, "Energy Efficient Bayesian Inference Using Near-Memory Computation with Memristors" , *Proceedings of the Design, Automation and Test in Europe Conference 2023*.

✠ A. Renaudineau, K.-E. Harabi, **C. Turck**, A. Laborieux, E. Vianello, M. Bocquet, J.-

M. PORTAL and D. QUERLIOZ, "Experimental Demonstration of Memristor DelayBased Logic In-Memory Ternary Neural Network" , *Proceedings of the Silicon Nanoelectronics Workshop 2023 (SNW 2023)*.

✠ MARIE DROUHIN, **CLÉMENT TURCK**, KAMEL-EDDINE HARABI, ADRIEN RENAUDINEAU, THOMAS BERSANI-VERONI, ELISA VIANELLO, JEAN-MICHEL PORTAL, JULIE GROLLIER and DAMIEN QUERLIOZ, "Nanoelectronic implementation of Equilibrium Propagation" , *Proceedings of Emerging Topics in Artificial Intelligence (ETAI) 2023.*.

✠ **C. TURCK**, D. BONNET, K.-E. HARABI, T. DALGATY, T. BALLET, T. HIRTZLIN, A. PONTLEVY, A. RENAUDINEAU, E. ESMANHOTTO, P. BESSIÈRE, J. DROULEZ, R. LAURENT, M. BOCQUET, J.-M. PORTAL, E. VIANELLO and D. QUERLIOZ, "Bayesian In-Memory Computing with Resistive Memories" , *69th Annual IEEE International Electron Devices Meeting*, 2023.

✠ HIRTZLIN TIFENN, DALGATY THOMAS, BOCQUET MARC, PORTAL JEAN-MICHEL, KLEIN JACQUES-OLIVIER, **TURCK CLÉMENT**, HARABI KAMEL-EDDINE, QUERLIOZ DAMIEN and VIANELLO ELISA, ""From resilience to Resistive Memory variability in Binarized Neural Networks to exploitation of variability in Bayesian Neural Network" , *2022 International Conference on IC Design and Technology (ICICDT)*. doi:10.1109/ICICDT56182.2022.9933076

Conferences Without Acts

✠ **CLÉMENT TURCK**, KAMEL-EDDINE HARABI, TIFENN HIRTZLIN, ELISA VIANELLO, MARC BOCQUET, JEAN-MICHEL PORTAL and DAMIEN QUERLIOZ, "Conception de systèmes efficaces en énergie dédiés à l'inférence bayésienne exploitant des nouvelles technologies mémoires" , *FETCH 2022 Workshop, Poster presentation*.

Appendix A

Annexe 1 : A Multimode Hybrid Memristor-CMOS Prototyping Platform Supporting Digital and Analog Projects

A Multimode Hybrid Memristor-CMOS Prototyping Platform Supporting Digital and Analog Projects

K.-E. Harabi, C. Turck
M. Drouhin, A. Renaudineau
T. Bersani--Veroni, D. Querlioz
damien.querlioz@c2n.upsaclay.fr
Univ. Paris-Saclay, CNRS
Palaiseau, France

T. Hirtzlin
E. Vianello
CEA-LETI, Univ. Grenoble-Alpes
Grenoble, France

M Bocquet
J.-M. Portal
Aix-Marseille Univ., CNRS
Marseille, France

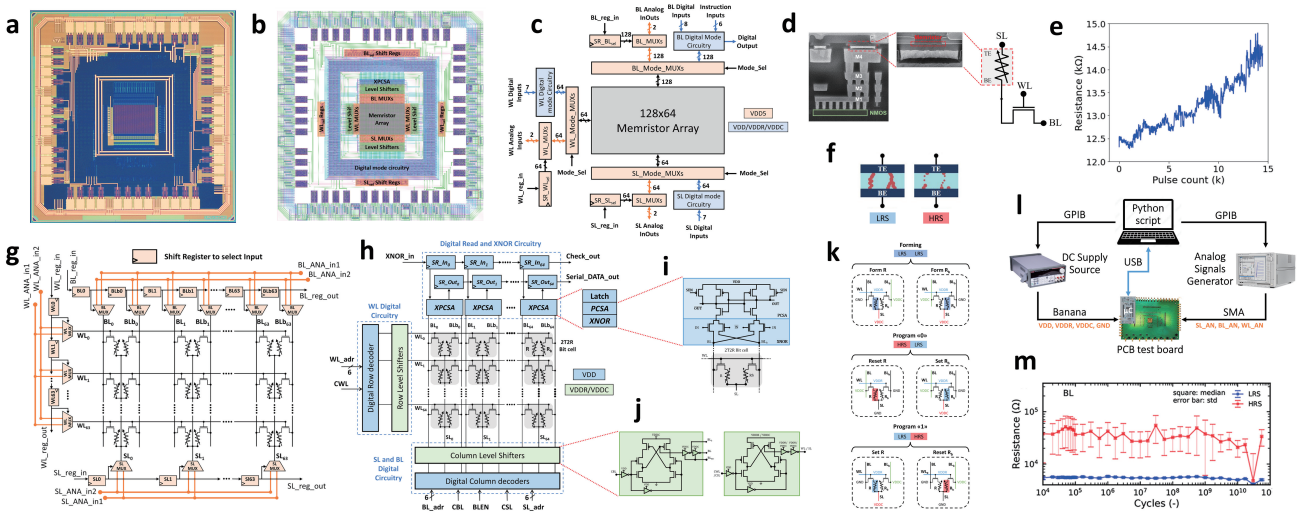


Figure 1: a Optical microscopy photograph, **b** layout view, and **c** schematic of the hybrid Memristor-CMOS die. **d** Electron microscopy image of a memristor in our hybrid memristor/CMOS process. **e** Measurement of memristor resistance as a function of number of RESET programming pulses, for implementing a synaptic learning rule. **f** Illustration of memristor programming states. **g** Schematic of the analog mode circuitry, with shift registers selecting inputs via Multiplexers . **h** Schematic of the digital mode circuitry, with a complementary 2T2R memristor basic cell. **i** Schematics of the sensing circuitry with XNOR logic-in-memory feature. **j** Schematic of the level shifters, used for shifting digital nominal voltage to forming and programming voltages of memristors. **k** Voltages applied for forming or programming a complementary cell in the digital mode. **l** Measurements setup of the prototyping platform. **m** Memristor endurance study, using the digital mode for programming and the analog mode for resistance measurements.

ABSTRACT

We present an integrated circuit fabricated in a process co-integrating CMOS and hafnium-oxide memristor technology, which provides a prototyping platform for projects involving memristors. Our circuit includes the periphery circuitry for using memristors within digital circuits, as well as an analog mode with direct access to memristors. The platform allows optimizing the conditions for reading and writing memristors, as well as developing and testing innovative memristor-based neuromorphic concepts.

CCS CONCEPTS

• **Hardware → Memory and dense storage**.

KEYWORDS

memristor, RRAM, prototyping platform, neural networks.

ACM Reference Format:
K.-E. Harabi, C. Turck, M. Drouhin, A. Renaudineau, T. Bersani--Veroni, D. Querlioz, T. Hirtzlin, E. Vianello, M Bocquet, and J.-M. Portal. 2023. A Multimode Hybrid Memristor-CMOS Prototyping Platform Supporting Digital and Analog Projects. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3566097.3567944

ASPDAC '23, January 16–19, 2023, Tokyo, Japan

ACM ISBN 978-1-4503-9783-4/23/01…$15.00
https://doi.org/10.1145/3566097.3567944

1 INTRODUCTION

Memristors, also known as resistive random access memories (RRAM) are a new type of memory technology fully embeddable in CMOS, providing a compact nonvolatile, and fast memory [6]. These devices provide fantastic opportunities to integrate logic and memory tightly and allow low-power computing, in particular for Artificial Intelligence models and neuromorphic computing [4]. Unfortunately, the behavior of memristors is highly complex and partly stochastic [3]: device models do not provide an accurate prediction of their behavior. It is therefore essential to prototype computing concepts involving memristors experimentally. However, appropriate platforms are extremely complex to fabricate due to the need to co-integrate commercial CMOS and memristor devices on the same die. In this work, we designed, fabricated, and tested a prototyping platform, associating an array of 8,192 hafnium-oxide-based memristors and a collection of CMOS periphery circuits. Our platform is multi-paradigm, which permits prototyping a wide range of both digital and analog projects.

2 DESCRIPTION OF THE DIE

A photograph and a layout view of our integrated circuit are presented in Figs. 1a-b. An electron microscopy image of a memristor in the backend of line of our hybrid memristor/CMOS process is shown in Fig. 1d. A commercial foundry fabricated the CMOS part (including the backend up to metal layer 4), using a 130-nanometer process. Afterward, we deposited the memristors on top of metal 4 using atomic layer deposition, and a fifth layer of metal.

Our integrated circuit embeds periphery circuitry enabling the use of memristors within two modes. Fig. 1c shows a simplified schematic of the circuit. It uses consistent color codes: blue-colored blocks are digital-mode circuits, designed using thin oxide low-power transistors and supplied by digital nominal voltage (except for level shifters), and orange-colored blocks are analog-mode circuits, designed using thick oxide transistors to be compatible with high voltages.

The digital mode circuits (Figs. 1h-k) consist of: row and column decoders to select devices based on input addresses, level shifters on each row and column, which shift digital nominal voltage to higher voltages required to form and program memristors, and precharge sense amplifiers, with a logic-in-memory feature, at each column [5]. These sense amplifiers allow reading the binary states of memory cells in a highly energy-efficient fashion while optionally performing logic operations at the same time. The complementary approach of [1] is used in our array for reducing the bit error rate.

When activating the analog mode (Figs. 1g), digital circuits are deactivated and the memristors array connections are switched to the analog circuitry. In this mode, shift registers configure input multiplexers permitting direct access to the analog state of memristors, using low-resistance transmission gates. Each word line, bit line, and source line is then connected to the ground or to one of two analog InOut Pads, which can be connected to external equipment, e.g., Keysight B1530, a pulse source and measurement unit widely used to characterize memory devices.

The memristor array and all analog and mixed-signal circuits were designed in a full custom fashion, based on an extensive work on memristor characterization, modeling, and simulation. All digital circuits were placed and routed automatically using an HDL description and a Cadence Encounter flow provided by the foundry. Then, all circuits of the system were assembled manually and routed automatically using a Cadence Encounter flow developed in-house using a homemade abstract view of the memory array.

3 USES OF THE PLATFORM

To make the system re-configurable for different projects, we developed the experimental setup of Fig. 1l: a PCB routes a microcontroller unit and measurement equipment with our packaged die. Python scripts control the measurement.

Optimizing read and programming strategies using the digital mode can allow the successful implementation of digital applications. Memristors feature a complex interplay between programming energy, reading speed, read disturb effects, and device endurance, which our platform allows understanding. Fig. 1m shows an endurance study example. A memristor is programmed repeatedly using the digital mode circuits, and the memristor resistance is checked regularly using the analog mode and reported in Fig. 1m. It shows that the memristor resistance starts to degrade after 10^9 cycles, concurrently with the emergence of bit errors seen by the reads after each programming using sense amplifiers (not shown in Figures). We observed that memristor endurance can vary between 10^3 and 10^9 cycles depending on programming conditions.

The analog mode of the platform can be used to prototype computing concepts where memristors are used in an analog fashion, e.g., as artificial synapses in machine learning or neuromorphic circuits [4]. Fig. 1e shows measurements on a memristor in our platform when applying a succession of 15,000 $1V$ 1.5-μs programming pulses: the memristor resistance progressively increases, a feature that permits the memristor to implement a synaptic learning rule. This use is particularly appealing due to its compactness, but the imperfections of memristors (thermal and random telegraph noise, cycle-to-cycle, and device-to-device variability) pose challenges that make it necessary to test ideas experimentally. Our platform supports prototyping various neuromorphic experiments, targeting inference, deterministic or probabilistic learning [2].

4 CONCLUSION

We have designed, fabricated, and tested a flexible multi-paradigm platform to prototype and optimize digital and/or analog computing concepts, based on a hybrid CMOS/memristor integrated circuit. We are currently using it to validate multiple digital logic-in-memory and analog neuromorphic concepts, and plan to make the platform available to other research groups.

ACKNOWLEDGMENTS

This work was supported by ERC Grant NANOINFER (715872) and ANR grant NEURONIC (ANR-18-CE24-0009).

REFERENCES

[1] M. Bocquet et al. 2018. In-Memory and Error-Immune Differential RRAM Implementation of Binarized Deep Neural Networks. In *IEDM Tech. Dig.* IEEE, 20.6.1.
[2] T. Dalgaty et al. 2021. In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling. *Nature Electronics* 4, 2 (2021), 151–161.
[3] A. Majumdar et al. 2021. Model of the Weak Reset Process in HfO x Resistive Memory for Deep Learning Frameworks. *IEEE Trans. Elect. Dev.* 68 (2021), 4925.
[4] S. Yu. 2018. Neuro-inspired computing with emerging nonvolatile memorys. *Proc. IEEE* 106, 2 (2018), 260–285.
[5] W. Zhao et al. 2014. Synchronous non-volatile logic gate design based on resistive switching memories. *IEEE TCAS I* 61, 2 (2014), 443–454.
[6] M A Zidan, J P Strachan, and W D Lu. 2018. The future of electronics based on memristive systems. *Nature electronics* 1, 1 (2018), 22–29.

Appendix B

Annexe 2 : Powering AI at the Edge: A Robust, Memristor-based Binarized Neural Network with Near-Memory Computing and Miniaturized Solar Cell

Preprint : arXiv:2305.12875

Powering AI at the Edge: A Robust, Memristor-based Binarized Neural Network with Near-Memory Computing and Miniaturized Solar Cell

Fadi Jebali[1], Atreya Majumdar[2], Clément Turck[2], Kamel-Eddine Harabi[2], Mathieu-Coumba Faye[1,4], Eloi Muhr[1], Jean-Pierre Walder[1], Oleksandr Bilousov[3], Amadéo Michaud[3], Elisa Vianello[4], Tifenn Hirtzlin[4], François Andrieu[4], Marc Bocquet[1], Stéphane Collin[2,3], Damien Querlioz[2,*], and Jean-Michel Portal[1,*]

[1]Aix-Marseille Université, CNRS, Institut Matériaux Microélectronique Nanosciences de Provence, Marseille, France.
[2]Université Paris-Saclay, CNRS, Centre de Nanosciences et de Nanotechnologies, Palaiseau, France.
[3]Institut Photovoltaïque d'Ile-de-France (IPVF), Palaiseau, France.
[4]Université Grenoble Alpes, CEA, LETI, Grenoble, France.
[*]damien.querlioz@c2n.upsaclay.fr, jean-michel.portal@univ-amu.fr

ABSTRACT

Memristor-based neural networks provide an exceptional energy-efficient platform for artificial intelligence (AI), presenting the possibility of self-powered operation when paired with energy harvesters. However, most memristor-based networks rely on analog in-memory computing, necessitating a stable and precise power supply, which is incompatible with the inherently unstable and unreliable energy harvesters. In this work, we fabricated a robust binarized neural network comprising 32,768 memristors, powered by a miniature wide-bandgap solar cell optimized for edge applications. Our circuit employs a resilient digital near-memory computing approach, featuring complementarily programmed memristors and logic-in-sense-amplifier. This design eliminates the need for compensation or calibration, operating effectively under diverse conditions. Under high illumination, the circuit achieves inference performance comparable to that of a lab bench power supply. In low illumination scenarios, it remains functional with slightly reduced accuracy, seamlessly transitioning to an approximate computing mode. Through image classification neural network simulations, we demonstrate that misclassified images under low illumination are primarily difficult-to-classify cases. Our approach lays the groundwork for self-powered AI and the creation of intelligent sensors for various applications in health, safety, and environment monitoring.

Introduction

Artificial intelligence (AI) has found widespread use in various embedded applications such as patient monitoring, building, and industrial safety[1]. To ensure security and minimize energy consumption due to communication, it is preferable to process data at the edge in such systems[2]. However, deploying AI in extreme-edge environments poses a challenge due to its high power consumption, often requiring AI to be relegated to the "cloud" or the "fog"[3,4]. A promising solution to this problem is the use of memristor-based systems, which can drastically reduce the energy consumption of AI[5,6], making it even conceivable to create self-powered edge AI systems that do not require batteries and can instead harvest energy from the environment. Additionally, memristors provide the advantage of being non-volatile memories, retaining stored information even if harvested energy is depleted.

The most-energy efficient memristor-based AI circuits rely on analog-based in-memory computing: they exploit Ohm's and Kirchhoff's laws to perform the fundamental operation of neural networks, multiply-and-accumulate (MAC)[7–9]. This concept is challenging to realize in practice due to the high variability of memristors, the imperfections of analog CMOS circuits, and voltage drop effects. To overcome these challenges, integrated memristor-based AI systems employ complex peripheral circuits, which are tuned for a particular supply voltage[10–16]. This requirement for a stable supply voltage is in direct contrast with the properties of miniature energy harvesters such as tiny solar cells or thermoelectric generators, which provide fluctuating voltage and energy, creating a significant obstacle to realizing self-powered memristor-based AI[17].

In this work, we demonstrate a binarized neural network, fabricated in a hybrid CMOS/memristor process, and designed with an alternative approach that is particularly resilient to unreliable power supply. We demonstrate this robustness by powering our circuit with a miniature wide-bandgap solar cell, optimized for indoor applications. Remarkably, the circuit maintains

functionality even under low illumination conditions equivalent to 0.08 suns, experiencing only a modest decline in neural network accuracy. When power availability is limited, our circuit seamlessly transitions from precise to approximate computing as it begins to encounter errors while reading difficult-to-read, imperfectly-programmed memristors.

Our fully digital circuit, devoid of the need for any analog-to-digital conversion, incorporates four arrays of 8,192 memristors each. It employs a logic-in-sense-amplifier two-transistor/two-memristor strategy for optimal robustness, introducing a practical realization of the near-memory computing concept initially proposed in ref.[18,19]. The design is reminiscent of the smaller-scale memristor-based Bayesian machine recently showcased in ref.[20], with the added novelty of logic-in-memory functionality. This feature is achieved by executing multiplication within a robust precharge differential sense amplifier, a circuit initially proposed in ref.[21]. Accumulation is then performed using a straightforward digital circuit situated near-memory. Our system also integrates on-chip a power management unit and a digital control unit, responsible for memristor programming and the execution of fully pipelined inference operations.

We first introduce our integrated circuit and provide a comprehensive analysis of its electrical characteristics and performance across a variety of supply voltages and frequencies. We then characterize the behavior of the circuit under solar cell power, demonstrating its adaptability and resilience even when the power supply is significantly degraded due to low illumination. To further showcase the robustness of the circuit, we present results from neural network simulations using the popular MNIST and CIFAR-10 datasets. These results highlight the capability of the circuit to perform well even under extremely low illumination conditions.

Results

Binarized neural network machine based on distributed memristor modules

In binarized neural networks, both synaptic weights and neuronal activations assume binary values (meaning $+1$ and -1)[22,23]. These networks are particularly appropriate for the extreme edge, as they can be trained for image and signal processing tasks with high accuracy, while requiring less resources than conventional real-valued neural networks[24,25]. In these simplified networks, multiplication can be implemented by a one-bit exclusive NOR (XNOR) operation and accumulation by a population count (popcount). The output neuron activations $X_{out,j}$ are, therefore, obtained by

$$X_{out,j} = \text{sign}\left(\text{popcount}\left(XNOR\left(W_{ji}, X_{in,i}\right)\right) - T_j\right), \tag{1}$$

using the synaptic weights W_{ji}, the input neuron activations $X_{in,i}$ and the output neuron threshold T_j. The quantity $\text{popcount}\left(XNOR\left(W_{ji}, X_{in,i}\right)\right) - T_j$ is a signed integer, referred to as neuron preactivation throughout this paper.

We fabricated a binarized neural network hardware system (Fig.s 1a,b) employing hafnium-oxide memristors integrated into the back end of a CMOS line to compute equation 1. The memristors replace vias between metal layers four and five (Fig. 1c) and are used to program the synaptic weights and neuron thresholds in a non-volatile manner. The system comprises four memristor arrays, each containing 8,192 memristors. These arrays can be used in two distinct configurations: one with two neural network layers featuring 116 inputs and 64 outputs, or an alternative single-layer configuration that has 116 inputs and 128 outputs. Additionally, we fabricated a smaller die that includes a single 8,192-memristor module with peripheral circuits that provide more flexibility to access memristors. Our circuits use a low-power 130-nanometer process node, which is interesting for extreme-edge applications, as it is cost-effective, offers well-balanced analog and digital performance, and supports a wide range of voltages. Due to the partially academic nature of our process, only five layers of metals are available.

Our design choices aim to ensure the most reliable operation under unreliable power supply and follow the differential strategy proposed in ref.[19]. To achieve this, we use two memristors per synaptic weight, programmed in a complementary fashion, with one in a low resistance state and the other in a high resistance state (see Fig. 1d). We also employ a dedicated logic-in-memory precharge sense amplifier[21] to perform the multiplication, which simultaneously reads the state of the two memristors representing the weight and performs an XNOR with its X input (Fig. 1f). This differential approach makes our circuit highly resilient. It minimizes the effects of memristor variability by ensuring that the sense amplifier functions as long as the memristor in the low resistance state has a lower resistance than the memristor in the high resistance state, even if they deviate significantly from their nominal values. Furthermore, fluctuations in the power supply voltage affect both branches of the sense amplifier symmetrically. This robustness eliminates the need for compensation and calibration circuits, unlike in other analog in-memory computing implementations that require a finely controlled supply voltage.

Our system computes the values of all output neurons in parallel. We provide a detailed description of the pipelined operation of the neural network in Supplementary Note 3, and summarize the main principle here. The neuron thresholds, which are stored in dedicated rows of the memristor arrays, are read simultaneously and transferred to neuron registers located near the memristor arrays. Then, input neurons are presented sequentially to the memristor array. The accumulation operation of the neural network is performed by integer digital population count circuits that take as input the outputs of the XNOR-augmented sense amplifiers and decrement the neuron registers. These circuits, which are replicated for each output neuron, are located

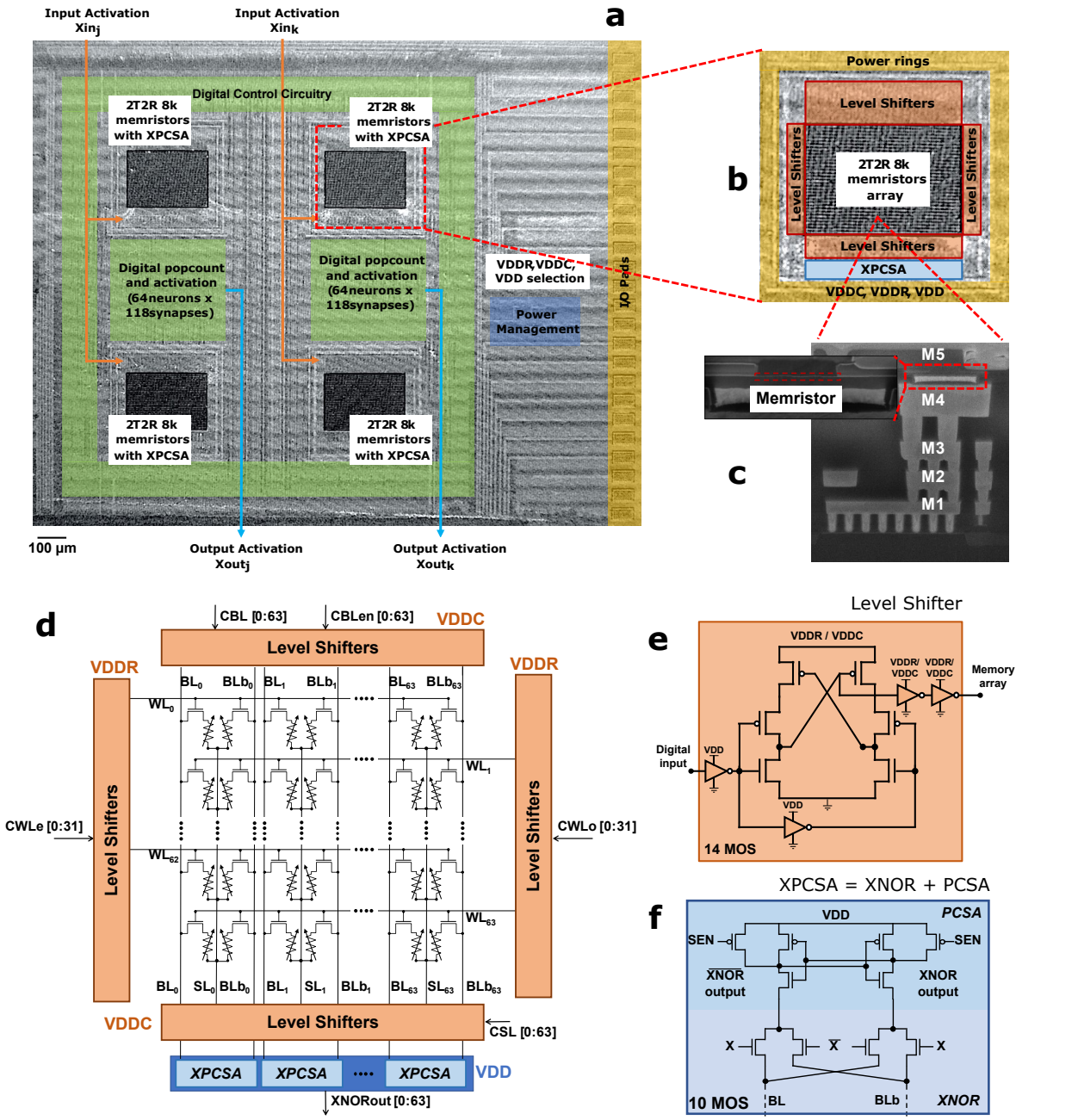


Figure 1. Overview of the fabricated memristor-based binarized neural network. a Optical microscopy image of the fabricated die, showing four memory modules and their associated digital circuitry and power management unit. **b** Detail on one of the memory modules. **c** Cross-sectional scanning electron micrograph of a hybrid CMOS/memristor circuit, showing a memristor between metal levels four and five. **d** Schematic of a memory module. For each operation mode, biasing conditions for WL, BL, and SL are given with respect to the power domain (VDDC, VDDR) and VDD. **e** Schematic of the level shifter, used in **d** for shifting digital voltage input to medium voltages needed during programming operations or nominal voltage during reading operations of the memristors. **f** Schematic of the differential pre-charge sense amplifier PCSA, used to read the binary memristor states, with embedded XNOR function, to compose a XPCSA: it computes an XNOR operation between input activation X and weight (memristor value) during bit-cell sensing.

physically near the memristor arrays. This near-memory computing principle saves energy, as only the binarized activations of the output neurons, obtained by taking the sign bit of the threshold register at the end of the inference process, need to be transmitted away from the memories.

As the synaptic weights are stored in non-volatile memory, the system can be turned off and on at any time, cutting power consumption completely, and can immediately perform a new inference or restart a failed one. The programming of the weights needs to be carried out prior to inference, and a forming operation must be performed on each memristor before its first programming operation. A challenge is that the forming operation requires voltages as high as 4.5 volts, whereas the nominal voltage of our CMOS process is only 1.2 volts. To overcome this, we included level shifters in the periphery circuitry of the memristor arrays (Fig. 1e), which can sustain high voltages. These circuits, similar to the ones used in ref.[20], use thick-oxide transistors to raise the voltage of the on-chip signals commanding the programming of memristors. The higher-than-nominal voltages are provided by two power pads. Once the memristors have been programmed, these pads can be connected to the digital low-voltage power supply VDD, as high voltages are no longer needed. The details of the memristor forming and programming operation are provided in Supplementary Note 2. Additionally, we incorporated a power management unit and a complete state machine into our fabricated circuit. These components, placed and routed all around the die, are detailed in Supplementary Note 1.

Characterization of the fabricated distributed memory modules BNN machine

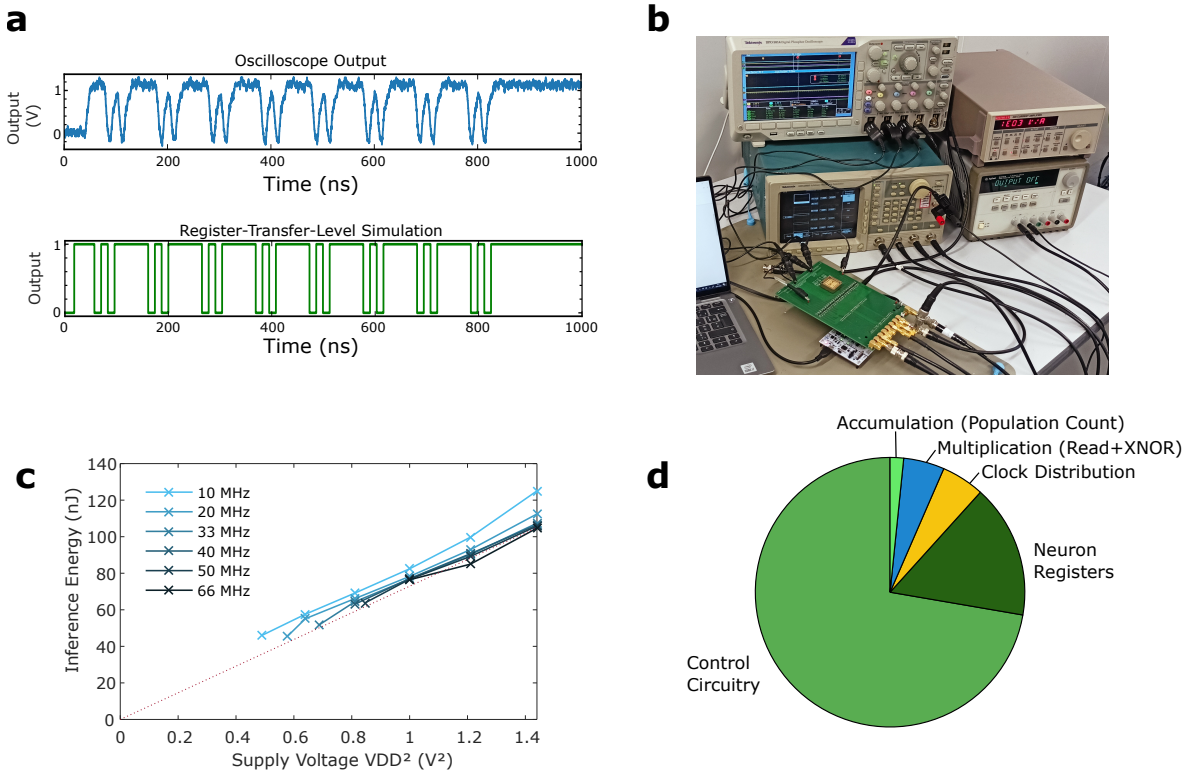


Figure 2. Measurements of the memristor-based binarized neural network, employing a lab-bench power supply. **a** Sample measurement of the output of the integrated circuit, compared with a delay-less register-transfer level (RTL) simulation. **b** Photograph of the printed circuit board used for the experiments. **c** Measurement of the energy consumption to perform a whole-chip inference, for various operating frequencies and supply voltages. **d** Pie chart comparing the different sources of energy consumption in the system, obtained using simulations (see Methods).

Our fabricated system is functional across a wide range of supply voltages and operating frequencies, without the need for calibration. As shown in Fig. 2a, the measured output of the system, obtained using the setup depicted in Fig. 2b, matches the register-transfer-level simulation of our design (see Methods). This first experiment was conducted using the maximum supported supply voltage of our process (1.2 volts) and a clock frequency of 66 MHz. The energy consumption of the system can be reduced by decreasing the supply voltage, as seen in Fig. 2c. This graph displays the measured energy consumption

across various supply voltages and frequencies where the system remained functional. The x-axis represents the square of the supply voltage to highlight its direct proportionality to energy consumption: all circuits on-chip, including the sense amplifiers, and with the exception of the power management circuits, function solely with capacitive loads. Notably, energy consumption is largely independent of operation frequency at a given supply voltage. This result, typical for CMOS digital circuits, suggests an absence of short-circuit currents in our design. Supply voltages lower than one volt do not support 66 MHz operation and require slower clock speeds. The lowest measured energy consumption of 45 nJ was achieved at a supply voltage of 0.7 volts (close to the threshold voltage of the transistors in the low-leakage process that we are using) and a clock frequency of 10 MHz.

Fig. 2d details the various sources of energy consumption in our circuit, as determined through simulations based on the process design kit of our technology. (It is not possible to separate the consumption of the different on-chip functions experimentally.) As the Figure illustrates, a significant portion of the energy is consumed by the on-chip digital control circuitry. In scaled-up systems, this proportion is expected to decrease considerably as the control circuitry would remain largely unchanged. Clock distribution represents only 5.2% of the energy, which is lower than typical digital circuits. This is due to the high proportion of circuit area taken up by memristor arrays, which do not require clock distribution. Neuron registers consume a substantial 16.0% of the energy, owing to their constant activity due to our design decision of not clock-gating them. This design choice simplified timing constraints in the circuit, ensuring its experimental functionality. However, a fully optimized design would be clock-gated, substantially reducing energy usage for the registers (see Discussion). The actual multiply-and-accumulate operations, including memristor read with XNOR logic-in-memory and population count, consume a modest 6.5% of the energy.

We now present a comprehensive characterization of the accuracy of our fabricated system. Initially, we programmed a memristor array with synaptic weights and neuron thresholds and tested it with neuron inputs, carefully selected to span the entire spectrum of potential output preactivation values (see Methods). Fig. 3a presents the measured accuracy (percentage of correct output neurons) across varying supply voltages and operational frequencies in a schmoo plot. With this setup, we observe no errors when the supply voltage is at least one volt. At 0.9 volts, occasional errors occur at 66 MHz operation, and below this voltage, error rates up to 2% can manifest at any frequency. We attribute these residual errors to the sense amplifiers, likely due to memristor variability and instability, which cause their resistance to deviate from the target nominal value. Conventional digital circuits incorporating memristors employ strong multiple-error correction codes to compensate for these issues[26]. By contrast, our sense amplifier, owing to its differential nature, can still determine the correct weight even if one memristor exhibits an improper resistance, as long as the memristor programmed in low resistance maintains a lower resistance than the memristor programmed in high resistance. At lower supply voltages, this task becomes more challenging, resulting in the observed residual bit errors.

As neuron errors arise from weight errors, they are only observed when the population count and threshold values of a neuron are comparable. We found that errors were absent experimentally when the difference between the population count and threshold (or neuron preactivation Δ) exceeded five. Figs. 3c,d, based on extensive experiments (see Methods), depict the error rates for different supply voltages as a function of the neuron preactivation, when the system operates at 33 MHz and 66 MHz. At a supply voltage of 1.2 volts, errors only occur when the preactivation is -1, 0, or 1. At a supply voltage of 0.9 volts, errors are observed for preactivation magnitudes up to five. To illustrate how errors occur, Fig. 3 shows measurements of 64 output neurons with varying preactivations values, ranging from -5 to +5, taken at 33 and 66 MHz, with a supply voltage of 0.9 volts. At this voltage, more errors are observed at 66 MHz than at 33 MHz. Almost all errors detected at 33 MHz continue to exist at 66 MHz. This observation implies that residual errors are likely due to specific weakly-programmed memristors (i.e., complementary memristors programmed with similar resistance), rather than random thermal noise.

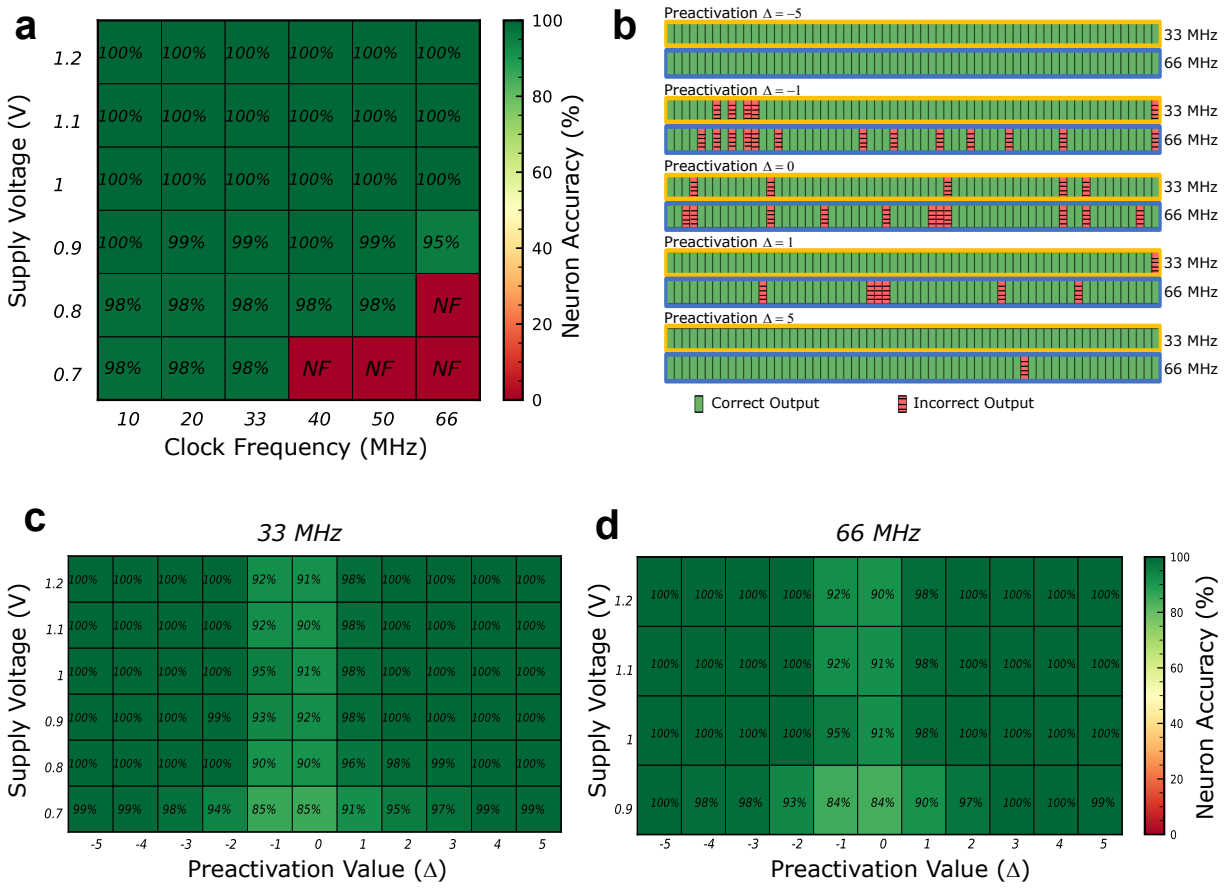


Figure 3. Accuracy of the memristor-based binarized neural network. a Measured schmoo plot, presenting mean accuracy of the output neuron activations, for different operation frequency and supply voltage. They were obtained using patterns of weights and inputs chosen to cover all possible neuron preactivations (see Methods). NF means non-functional. **b** Measurements of 64 neurons with preactivations -5, -1, 0, 1, and 5, at 33 and 66 MHz with a power supply of 0.9 volts. Errors are marked in red. **c,d** Mean accuracy of the output neuron activations, as a function of neuron preactivation Δ and supply voltage, measured at (**c**) 33 and (**d**) 66 MHz (see Methods).

Powering the system with harvested energy

To validate the suitability of our circuit for energy harvesting applications, we connected it to a miniature AlGaAs/GaInP heterostructure solar cell (see Fig. 4a and Methods). Fig. 4b displays a photograph of this cell, along with its current-voltage characteristics measured under standardized one-sun AM1.5 illumination (see Methods). This type of solar cell, fabricated following the procedure of ref.[27] (see Methods), with a 1.73 eV bandgap, performs better than conventional silicon-based cells under low-illumination conditions, making it particularly suitable for extreme edge applications. Additionally, due to the wide bandgap, the open-circuit voltage provided by our solar cell (1.23 volts under high illumination) aligns with the nominal supply voltage of our CMOS technology (1.2 volts), unlike silicon solar cells, whose maximum voltage is only 0.7 volts.

While energy harvesters are typically connected to electronic circuits through intricate voltage conversion and regulation circuits, we demonstrate the resilience of our binarized neural network by directly connecting the power supply pads of our circuit to the solar cell, without any interface circuitry. In those experiments, the solar cell is illuminated by a halogen lamp (Fig. 4d). Fig. 4c presents the current voltage of the solar cell with this setup for various illuminations, expressed as "equivalent solar powers" based on the short-circuit current of the solar cell (see Methods). Fig. 4e shows the measured accuracy of our system, plotted as a function of neuron preactivation, similarly to Fig. 3d.

Under an equivalent solar power of 8 suns, the circuit performs almost equivalently to when powered by a 1.2 volts lab bench supply. When illumination decreases, even under a very low equivalent solar power of 0.08 suns where the characteristics of the solar cell is strongly degraded, the circuit remains functional. However, its error rate increases, especially for low-magnitude preactivation values. The circuit naturally transitions to an approximative computing regime: neurons will large-magnitude preactivations are correctly computed, but those with low-magnitude preactivations may exhibit errors.

Equivalent Solar Power	MNIST Accuracy	CIFAR-10 Accuracy
Baseline	97.2%	86.6%
8 suns	97.1%	83.6%
0.8 suns	96.9%	78.2%
0.36 suns	96.9%	78.3%
0.08 suns	96.5%	73.4%

Table 1. Simulated accuracy of solar-cell power a fully-connected (MNIST task) and a convolutional (CIFAR-10 task) binarized neural network under various illuminations. The software baseline assumes no bit error (see Methods).

We now evaluate the performance of our circuit on neural networks. Our system functions with 128×64 memristor arrays; however, in practice, neural networks can have various structures. To map neural networks to our hardware, we employ a technique that subdivides neural network layers into several binarized arrays and then obtains the value of output neurons through majority votes of the binary output of each array (see Figs. 5a,b). This method, which we describe in more detail in Supplementary Note 4, is highly efficient in terms of hardware usage and causes only moderate accuracy degradation compared to software-based neural networks on the two tasks considered here: Modified National Institute of Standards and Technology (MNIST) handwritten digit recognition and CIFAR-10 image recognition.

To evaluate the classification accuracy of our hardware, we incorporated the error rates measured experimentally as a function of preactivation value and illumination (Fig. 4d) into neural network simulations (see Methods). Table 1 lists the obtained accuracy on a fully-connected neural network trained on MNIST and a convolutional neural network trained on CIFAR-10 (see Methods). Remarkably, the MNIST accuracy is hardly affected by the bit errors in the circuit: even under very low illumination equivalent to 0.08 suns, the MNIST accuracy drops by only 0.7 percentage points. Conversely, bit errors significantly reduce the accuracy of the more demanding CIFAR-10 task. Under 0.08 suns, the accuracy drops from the software baseline of 86.6% to 73.4%. The difference with the MNIST arises because more neurons tend to have low-magnitude preactivation when solving CIFAR-10, as the differences between classes are more subtle.

To further understand the impact of low illumination on neural network performance, we plotted the t-distributed stochastic neighbor embedding[28] (t-SNE) representation of the MNIST test dataset in Fig. 5b. This technique represents each image as a point in a two-dimensional space, where similar images cluster together and dissimilar ones reside at a distance. In the left image, we marked in black the images that were correctly classified by a neural network under illumination equivalent to 8 suns, but incorrectly under 0.8 suns. Interestingly, these images tend to be on the edges of the clusters corresponding to the different digit classes, or even outliers that do not belong in a cluster. This suggests that the images that the network starts misclassifying under 0.8 suns tend to be subtle or atypical cases. The right image shows that this effect intensifies under illumination equivalent to 0.08 suns, with a few images inside clusters also being misclassified. Fig. 5c presents the same analysis for the CIFAR-10 dataset. The trend of incorrectly classified images under low illumination tending to be edge or atypical cases persists, albeit less pronounced than with MNIST.

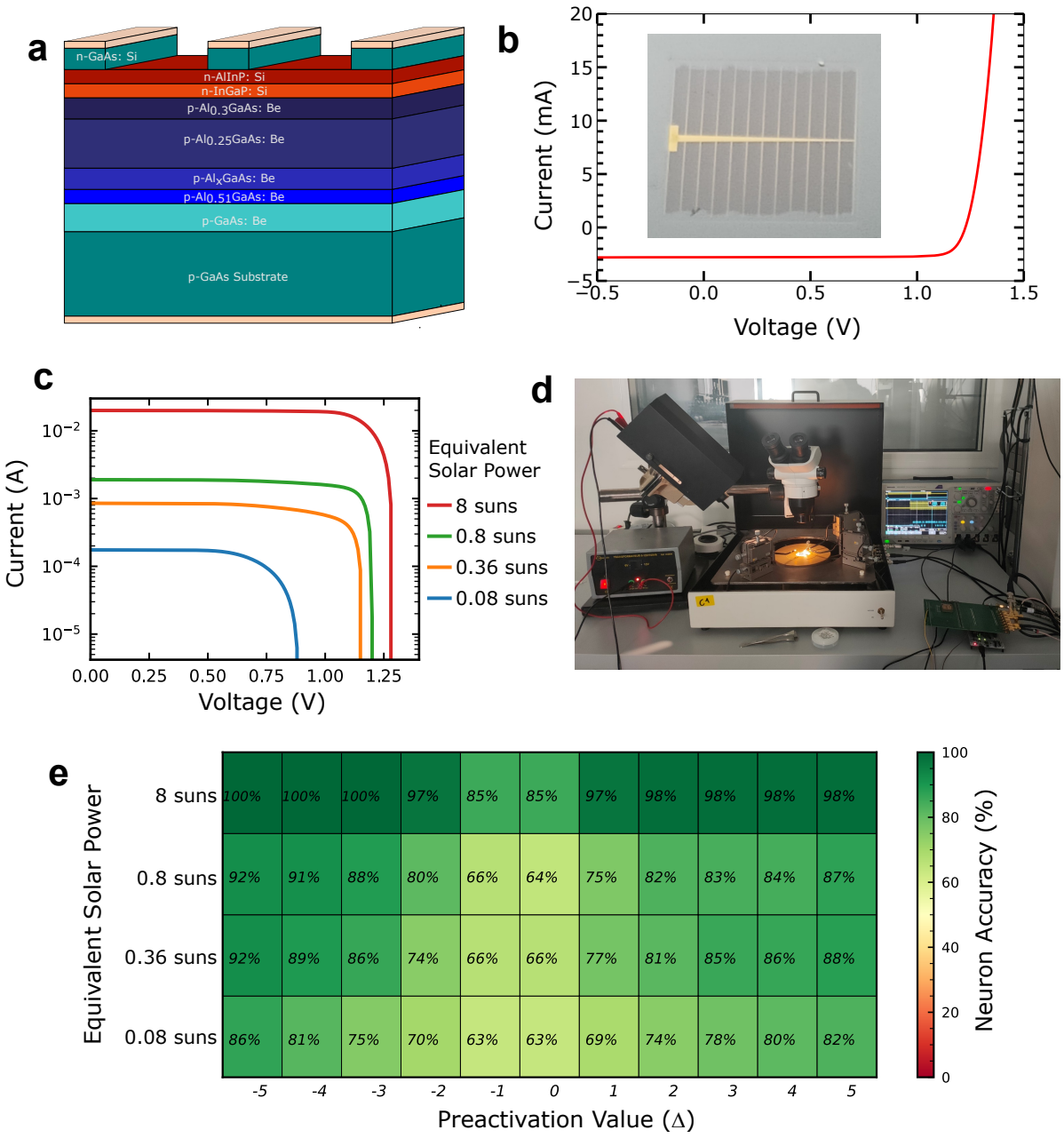


Figure 4. Measurements of the binarized neural network powered by a miniature solar cell. a Schematic view of the AlGaAs/GaInP heterostructure solar cell. **b** Photograph of the solar cell, and its measured current-voltage characteristics under one-sun AM1.5 illumination provided by a standardized solar simulator (see Methods). **c** Current-voltage characteristics of the solar cell for various illuminations provided by the halogen lamp (see Methods). **d** Photograph of the experimental setup where the fabricated binarized neural network is powered by the solar cell illuminated by the halogen lamp. **e** Mean measured accuracy of the output neuron activations, with the binarized neural network powered by the solar cell, as a function of neuron preactivation Δ and solar cell illumination.

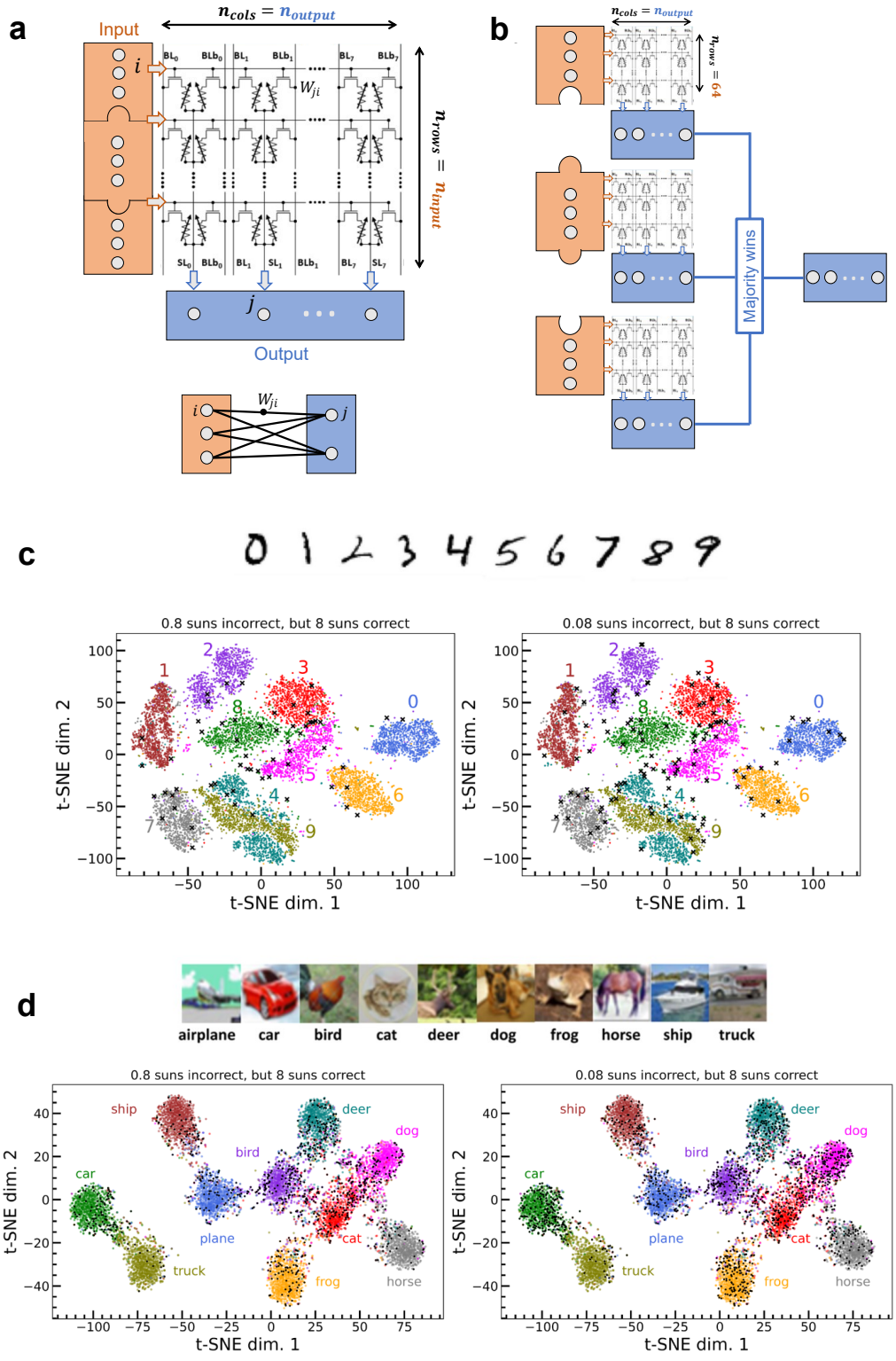


Figure 5. Neural-network-level investigations. a,b Illustration of our method for mapping arbitrarily-shaped binarized neural networks to 64×128 memristor arrays. The detailed methodology is presented in Supplementary Note 4. **c** t-distributed stochastic neighbor embedding (t-SNE) representation of the MNIST test dataset. The datapoints incorrectly classified under 0.8 suns (left) and 0.08 suns (right) equivalent illumination, but which would be correctly classified under 8 suns, are marked in black. These results are obtained using a binarized fully-connected neural network (see Methods). **d** Same graphs as **c**, obtained for the CIFAR-10 dataset and using a convolutional neural network (see Methods).

Discussion

Our circuit exhibits an original behavior when solving tasks of varying difficulty levels. For simpler tasks such as MNIST, the circuit maintains accuracy even when energy is scarce. When addressing more complex tasks, the circuit becomes less accurate as energy availability decreases, but without failing completely. This self-adaptive approximate computing feature has several roots and can be understood by the circuit's memory read operations. They are highly robust due to their differential nature: fluctuations of the power supply affect both branches of the sense amplifier equally. Still, when power voltage fluctuates or becomes low, some memory reads fail. Nevertheless, binarized neural networks are highly robust to weight errors, which in many cases do not change neuron activation[29,30]. Even in the worst case, weight errors cause some images to be misclassified, but these are typically atypical or edge cases. Therefore, when the power supply degrades, the AI naturally becomes less capable of recognizing harder-to-classify images.

In this context of low-quality power supply, memristors offer distinct advantages over conventional static RAMs. While static RAMs lose stored information upon power loss, memristors retain data. Furthermore, when the supply voltage becomes low, static RAMs are prone to read disturb, meaning that a read operation can change the bit stored in a memory cell. In contrast, memristors exhibit near-immunity to read disturb effects, especially when read by precharge sense amplifiers[20] (we observed no read disturb in our experiments), and are non-volatile (ten-years retention has been demonstrated in hafnium-oxide memristors[31]).

After eliminating the energy used by the digital control circuitry (finite state machine), our circuit has an energy efficiency of 2.9 tera-operations per second and per watt (TOPS/W) under optimal conditions (10 MHz frequency, supply voltage of 0.7 volts). By further subtracting the energy consumption of clock distribution and neuron registers that can be eliminated through clock gating, and simultaneously optimizing the read operation (see Methods), energy efficiency increases to 22.5 TOPS/W. Due to the digital nature of our circuit, this number would scale favorably if a more current CMOS process was used. For example, employing the physical design kit of a fully-depleted silicon-on-insulator 28-nanometer CMOS process, we found that the energy efficiency of a clock-gated design would reach 397 TOPS/W (see Methods). Supplementary Note 5 compares these numbers and other properties of our digital system with fabricated emerging memory-based analog in-memory computing circuits. The most noteworthy comparison is with a recent study that presents an analog magnetoresistive memory (MRAM) based 64x64 binarized neural network fabricated in a 28-nanometer process[14], which has a measured energy efficiency of 405 TOPS/W, which surpasses our projection slighly. However, this energy efficiency comes with the need for complex compensation and calibration circuits, matched to a stable power supply, which is not suitable with the unreliable power supply delivered by energy harvesters.

Our circuit can function with power supplies as low as 0.7 volts, enabling us to power it with a wide-bandgap solar cell optimized for indoor applications, with an area of only a few square millimeters, even under low illumination equivalent to 0.08 suns. Such lightweight, ultrathin solar cells can also be transferred into a fully-integrated, self-powered device[32,33]. Supply voltages lower than 0.7 volts result in significant inaccuracies in memristor readings due to the high threshold voltages of the thick-oxide transistors in our process. Employing a process with a lower threshold voltage thick-oxide transistor option could enable operation at lower supply voltages, broadening compatibility with various solar and non-solar energy harvesters. Some very low-voltage harvesters (e.g., thermoelectrics) may still require the voltage to be raised, which can be accomplished on-chip using switched capacitor circuits like Dickson charge pumps[34]. Self-powered AI at the edge, therefore, offers multiple opportunities to enable the development of intelligent sensors for health, safety, and environmental monitoring.

Acknowledgements

This work received funding within the ECSEL Joint Undertaking (JU) project storAIge in collaboration with the European Union's H2020 research and innovation program and National Authorities, under grant agreement numbers 101007321. This work was also supported by European Research Council starting grant NANOINFER (reference: 715872), by the Agence Nationale de la Recherche through the NEURONIC (ANR-18-CE24-0009) grant, by the French Government in the framework of the "Programme d'Investissement d'Avenir" (ANR-IEED-002-01), and with the support of the cleanroom RENATECH network. It also benefits from a France 2030 government grant managed by the French National Research Agency (ANR-22-PEEL-0010). The authors would like to thank J. Grollier and L. Hutin for discussion and invaluable feedback. Parts of this manuscript were revised with the assistance of a large language model (OpenAI ChatGPT).

Author contributions statement

J.M.P designed the hardware binarized neural network, using a flow developed with J.P.W., and with contributions from M.C.F., E.M., and D.Q. The system was fabricated under the direction of E.V. and F.A. F.J. performed the on-chip experimental measurements, under the direction of M.B. C.T., and K.E.H. analyzed the energy consumption of the system. A.M. performed the neural network-level analyses. O.B., A.M., and S.C. developed and fabricated the solar cells. T.H. developed the binarized

neural network simulator. D.Q. and J.M.P. directed the project and wrote the initial version of the manuscript. All authors discussed the results and reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Data availability

The datasets analyzed and all data measured in this study are available from the corresponding author upon reasonable request.

Code availability

The software programs used for modeling the Binarized Neural Network machine are available from the corresponding author upon reasonable request.

Methods

Fabrication of the demonstrator

The MOS part of our demonstrator was fabricated using a low-power 130-nanometer foundry process up to the fourth layer of metal. Memristors, composed of a TiN/HfO_x/Ti/TiN stack, were then fabricated on top of exposed vias. The active 10-nanometer thick HfO_x layer was deposited by atomic layer deposition. The Ti layer is also 10-nanometer thick, and the memristor structure has a diameter of 300 nanometers. A fifth layer of metal was deposited on top of the memristors. 25 input/output pads are aligned to be compatible with a custom probe card. A packaged version of the demonstrator was also assembled in a J-Leaded Ceramic Chip Carrier with 52 leads.

Design of the demonstrator

The memristor-based Binarized Neural Network is a hybrid CMOS/nanotechnology integrated circuit with distributed memory modules within the logic. The design of the memory module includes the array and peripheral circuits, such as the XNOR-augmented precharge sense amplifiers (Fig. 1b) and the level shifter circuits (Fig. 1c). The memory modules were designed using a full-custom flow under the Cadence Virtuoso electronic design automation (EDA) tool and were simulated using the Siemens Eldo simulator. Verification steps, i.e., layout versus schematic check and design rule check, were performed using Calibre tools.

The level shifter circuit (Fig. 1e) was designed with thick-oxide MOS transistors supporting up to five volts. To isolate the precharge current sense amplifier during the forming or programming operations, the four XNOR MOS transistors (the ones connected to the input X in Fig. 1f) were designed with thick gate oxide. The sense amplifier itself was constructed using thin gate oxide transistors. The memory modules architecture also includes four dedicated power rings: one for VDDR, one for VDDC, one for VDD, and one for the ground (GND). An abstract view of the memory modules was generated using the Cadence abstract generator. The power switch unit, which has to sustain up to 4.5 volts during the forming operation, was also designed using thick-oxide transistors, following the same full-custom flow as the memory modules. A Liberty Timing Files (.lib) related to the abstract view of the full custom blocks was handwritten and a Synopsys database file (.sdb) was generated using the Synopys Library Compiler.

The overall machine core follows a digital on-top flow, where all digital blocks (e.g., controller logic, population count decounter, neuron registers) are described using the VHSIC Hardware Description Language (VHDL), including the full custom blocks entity, synthesized using the Synopsys Design Compiler, and finally placed and routed, including the full-custom abstract view, using the Cadence Encounter RTL-to-GDSII tool, following a semi-automated flow developed by the foundry. All digital circuits use thin-oxide high-threshold transistors and are biased to VDD. Logical verification of the core, including the memory modules, described with an equivalent handmade VHDL behavioral description, and the power switch, described with an equivalent handmade VerilogA description, were performed using Siemens Questa mixed-signal simulator. The memory modules equivalent VHDL description and the power switch VerilogA equivalent descriptions were first assessed against their electrical schematic counterparts, simulated with Siemens Eldo electrical simulator. The connection of the machine layout to the 25 input/output pads was accomplished manually in a full-custom fashion.

Supplementary Note 1 describes the digital control circuitry and the power management unit with more technical details. Supplementary Note 2 details the methodology used by our circuit for forming and programming the memristors. Supplementary Note 3 lists the steps of the pipelined inference operation of the circuit.

Fabrication of the solar cell

The fabrication of solar cells in this study was carried out according to the procedures described in ref.[27]. The semiconductor stack was grown on a GaAs substrate using molecular beam epitaxy and consisted of the following sequence of layers: p-GaAs:Be (300 nm), p-$Al_{0.51}$GaAs:Be (50 nm), p-AlGaAs:Be with a linear gradient from 51% to 25% Al (100 nm), p-$Al_{0.25}$GaAs:Be (1900 nm), n-$Al_{0.3}$GaAs:Si (100 nm), n-InGaP:Si (50 nm), n-AlInP:Si (20 nm), and n-GaAs:Si (300 nm) (see Fig. 4a).

The front metal grid was defined using standard photolithography techniques, followed by metal evaporation (NiGeAu) and lift-off processes. Wet chemical etching was used to separate the mesa structures of the different cells, and to etch the top 300 nm-thick GaAs contact layer outside the front grid area. The back contact (TiAu) is deposited on the backside of the substrate. No anti-reflection coating was added. The size of the solar cells is 5 mm $\times$5 mm.

Measurements of the system with lab-bench power supply

The measurements of our system were conducted on the packaged version. The binarized neural network integrated circuit is mounted on a dedicated printed circuit board (PCB) featuring level shifters and SubMiniature A (SMA) connectors (see Fig. 2b). The PCB connects the different input and output signals of the packaged chip to an STM32F746ZGT6 microcontroller unit, a Tektronix AWG2005 arbitrary waveform generator, and a Tektronix DPO 3014 oscilloscope. The voltage for the level shifters of the PCB is supplied by an Agilent E3631A power supply. The microcontroller unit is connected to a computer using a serial connection, while lab-bench equipments are connected to the computer using a National Instruments GPIB connection. The whole setup is controlled using python within a single Jupyter notebook.

Supplementary Note 2 details the memristor forming and programming operations, and we summarize them here. Before starting any measurement, all the memristors are formed, sequentially, under the control of the on-chip digital control block (Supplementary Note 1). During this operation, the VDDC supply voltage is set to 4.5 volts, VDDR to 2.7 volts, and VDD to 1.2 volts, during ten microseconds. After this initial forming step, the memristor array is programmed with the desired pattern (synaptic weights and neuron thresholds). The programmed data are transmitted to the microcontroller unit, which sends them to the binarized neural network integrated circuit row-by-row. To program a memory cell to HRS, the digital control block connects VDDC to 2.7 volts and VDDR to 4.5 volts, with VDD fixed at 1.2 volts. To program a memory cell to LRS, VDDC, and VDDR are both connected to 2.7 volts, and VDD is connected to 1.2 volts. The two memristors of each bit cell are always programmed in a complementary fashion (i.e., either LRS/HRS or HRS/LRS). The digital circuitry controls the programming operations based on the weight value for each memristor, and applies the programming pulses during six microseconds.

To perform inference (see Supplementary Note 3), input neuron activations are sent through the microcontroller unit for each row, and the output of the integrated circuit is captured by the microcontroller unit and stored in a comma-separated values (CSV) file. To obtain the schmoo plots shown in Fig. 3, the power supply voltage VDD was varied from 1.2 to 0.7 volts, for all considered operation frequencies. The saved outputs of the integrated circuit were compared to the expected outputs to extract the system's accuracy.

To measure the power consumption of the circuit (Fig. 2c), the VDD power supply of the test chip is connected to a Keithley 428 current amplifier. The output of the Keithley 428 is connected to the oscilloscope to obtain the current during inference.

Measurements of the system powered by the solar cell

We first characterized the current-voltage characteristics of the solar cell (Fig. 4b), using a certified solar simulator providing a one-sun (100 mW/cm^2) AM1.5 illumination. To power our binarized neural network by the solar cell, we switched to a more accessible variable-illumination halogen lamp (Fig. 4c), whose spectrum does not match AM1.5 solar light. To obtain an equivalent solar power, we measured the current-voltage characteristics of the solar cell under this lamp (Fig. 4b) using the source measure unit mode of a Keysight B1530A unit. We calculate the equivalent solar power by dividing the short circuit current by the one under one-sun AM1.5 illumination.

We then directly connected our binarized neural network to the solar cell and conducted inference measurements using the same methodology as with the lab-bench power supply. To accomplish this, we connected all three power pads of the circuit (VDD, VH, VM, see Supplementary Note 1) to the solar cell, as high supply voltages are not needed to perform inference. To obtain the schmoo of Fig. 4e, we used the same methodology as for Fig. 3, but by varying the halogen light illumination instead of the bench power supply voltage.

Energy consumption estimates

Energy measurements of the system (shown in Fig. 2c) cannot differentiate the consumption of the different elements of the circuit, as they all share the same power supply. To overcome this limitation (as illustrated in Fig. 2s), we relied on computer simulations of our circuit using commercial integrated circuit design tools.

We obtained energy estimates during the inference phase, after the memristors were formed and the memory programmed. The consumption of the memristor arrays was determined using circuit simulations (based on the Siemens Eldo simulator),

which also accounted for parasitic capacitance extracted from the memristor array layout. For the remainder of the system, we analyzed it using the Cadence Voltus power integrity solution framework on the placed-and-routed design, incorporating all parasitics. We utilized a value change dump (VCD) file obtained from a test bench simulation to ensure a realistic situation.

The memristor array blocks are full-custom and, therefore, not included in the standard library of the foundry. This raised a concern regarding the continuous flow of values before and after the memristor array when performing the energy analysis. To address this, we wrote a new liberty (.lib) file, specifically for use in the energy analysis, based on the actual output values during simulation to ensure that the flow before and after the memory was respected during the inference phase.

In our fabricated circuit, the neuron registers are enabled when an XNOR-augmented sense operation is performed. We chose not to clock-gate these registers to avoid any timing risk in our test chip; however, this strategy can be employed to reduce the energy consumption of a final design. Therefore, we also designed a clock-gated version of our circuit and estimated its energy consumption using the same flow as for the fabricated version. This clock-gated version also uses an optimized read process requiring fewer clock cycles. We finally estimated the energy consumption of a scaled-down version of the design in a commercial 28-nanometer fully-depleted silicon-on-insulator CMOS design kit. For this analysis, the memristor array was entirely redesigned in the 28-nm design kit. For the digital part, we use a scaling factor relating the typical energy consumption of equivalent circuits in the two commercial technology nodes.

Neural-network level investigations

For the neural network simulations presented in Fig. 2, we used a fully connected architecture for the MNIST handwritten digit recognition task, and a convolutional neural network architecture for the CIFAR-10 image classification task. Except for the input to the first layer, the activations and weights of the network were binarized, following the binarized neural network implementation[23]. The fully connected (FC) network had two hidden layers with 1,102 and 64 neurons, whereas the convolutional architecture was based on the VGG-16 network, and it consisted of 3x3 kernels for convolutions (Conv), batch normalizations (BN), and nxn for MaxPool (MPn) and reads: [Conv 198, BN, Conv 198, MP 2, BN, Conv 354, BN, Conv 354, MP2, BN, Conv 738, BN, Conv 406, MP3, FC(1102-1102-10)]. The number of hidden layer units and convolutional filters were chosen in accordance with the dedicated mapping technique described in Supplementary Note 4, such that the total number of blocks is always odd when a block size of 58 is used.

We trained the networks without errors and with the mapping technique implemented. The input neurons of the first layer and the output neurons of the final layer are non-binary, so we did not include circuit-induced errors in these layers, as they require different circuits. The convolutional network was trained for 500 epochs with the Adam optimizer with weight decay and a cosine annealing learning rate scheduler. The fully-connected network was trained with the same optimizer for 200 epochs with a step learning rate scheduler[35]. Only after the training was completed were the errors introduced during the inference step, using a dedicated Pytorch code reproducing the error rate measured experimentally (Fig. 4e). The error rate of the circuit for a certain level of illumination and a certain preactivation Δ was taken as the probability of having an error in the neuronal output. The PyTorch deep learning framework was used to perform all the neural network simulations.

References

1. Cui, L., Yang, S., Chen, F., Ming, Z., Lu, N. & Qin, J. A survey on application of machine learning for internet of things. *Int. J. Mach. Learn. Cybern.* **9**, 1399–1417 (2018).
2. Warden, P. & Situnayake, D. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers* (O'Reilly Media, 2019).
3. Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M. & Liljeberg, P. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Futur. Gener. Comput. Syst.* **78**, 641–658 (2018).
4. Qadri, Y. A., Nauman, A., Zikria, Y. B., Vasilakos, A. V. & Kim, S. W. The future of healthcare internet of things: a survey of emerging technologies. *IEEE Commun. Surv. & Tutorials* **22**, 1121–1167 (2020).
5. Yu, S. Neuro-inspired computing with emerging nonvolatile memorys. *Proc. IEEE* **106**, 260–285 (2018).
6. Ielmini, D. & Wong, H.-S. P. In-memory computing with resistive switching devices. *Nat. Electron.* **1**, 333 (2018).
7. Ambrogio, S., Narayanan, P., Tsai, H., Shelby, R. M., Boybat, I., Nolfo, C., Sidler, S., Giordano, M., Bodini, M., Farinha, N. C. *et al.* Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **558**, 60 (2018).
8. Prezioso, M., Merrikh-Bayat, F., Hoskins, B., Adam, G. C., Likharev, K. K. & Strukov, D. B. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61 (2015).
9. Wang, Z., Joshi, S., Savel'ev, S., Song, W., Midya, R., Li, Y., Rao, M., Yan, P., Asapu, S., Zhuo, Y. *et al.* Fully memristive neural networks for pattern classification with unsupervised learning. *Nat. Electron.* **1**, 137 (2018).

10. Xue, C.-X., Chiu, Y.-C., Liu, T.-W., Huang, T.-Y., Liu, J.-S., Chang, T.-W., Kao, H.-Y., Wang, J.-H., Wei, S.-Y., Lee, C.-Y. *et al.* A cmos-integrated compute-in-memory macro based on resistive random-access memory for ai edge devices. *Nat. Electron.* **4**, 81–90 (2021).

11. Li, C., Ignowski, J., Sheng, X., Wessel, R., Jaffe, B., Ingemi, J., Graves, C. & Strachan, J. P. Cmos-integrated nanoscale memristive crossbars for cnn and optimization acceleration. In *2020 IEEE International Memory Workshop (IMW)*, 1–4 (IEEE, 2020).

12. Yao, P., Wu, H., Gao, B., Tang, J., Zhang, Q., Zhang, W., Yang, J. J. & Qian, H. Fully hardware-implemented memristor convolutional neural network. *Nature* **577**, 641–646 (2020).

13. Wan, W., Kubendran, R., Eryilmaz, S. B., Zhang, W., Liao, Y., Wu, D., Deiss, S., Gao, B., Raina, P., Joshi, S. *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)*, 498–500 (IEEE, 2020).

14. Jung, S., Lee, H., Myung, S., Kim, H., Yoon, S. K., Kwon, S.-W., Ju, Y., Kim, M., Yi, W., Han, S. *et al.* A crossbar array of magnetoresistive memory devices for in-memory computing. *Nature* **601**, 211–216 (2022).

15. Khaddam-Aljameh, R., Stanisavljevic, M., Mas, J. F., Karunaratne, G., Brändli, M., Liu, F., Singh, A., Müller, S. M., Egger, U., Petropoulos, A. *et al.* Hermes-core—a 1.59-tops/mm 2 pcm on 14-nm cmos in-memory compute core using 300-ps/lsb linearized cco-based adcs. *IEEE J. Solid-State Circuits* **57**, 1027–1038 (2022).

16. Wan, W., Kubendran, R., Schaefer, C., Eryilmaz, S. B., Zhang, W., Wu, D., Deiss, S., Raina, P., Qian, H., Gao, B. *et al.* A compute-in-memory chip based on resistive random-access memory. *Nature* **608**, 504–512 (2022).

17. Ku, M.-L., Li, W., Chen, Y. & Liu, K. R. Advances in energy harvesting communications: Past, present, and future challenges. *IEEE Commun. Surv. & Tutorials* **18**, 1384–1412 (2015).

18. Bocquet, M., Hirztlin, T., Klein, J.-O., Nowak, E., Vianello, E., Portal, J.-M. & Querlioz, D. In-memory and error-immune differential rram implementation of binarized deep neural networks. In *IEDM Tech. Dig.*, 20.6.1 (IEEE, 2018).

19. Hirtzlin, T., Bocquet, M., Penkovsky, B., Klein, J.-O., Nowak, E., Vianello, E., Portal, J.-M. & Querlioz, D. Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays. *Front. neuroscience* **13**, 1383 (2020).

20. Harabi, K.-E., Hirtzlin, T., Turck, C., Vianello, E., Laurent, R., Droulez, J., Bessière, P., Portal, J.-M., Bocquet, M. & Querlioz, D. A memristor-based bayesian machine. *Nat. Electron.* 1–12 (2022).

21. Zhao, W., Moreau, M., Deng, E., Zhang, Y., Portal, J.-M., Klein, J.-O., Bocquet, M., Aziza, H., Deleruyelle, D., Muller, C. *et al.* Synchronous non-volatile logic gate design based on resistive switching memories. *IEEE Transactions on Circuits Syst. I: Regul. Pap.* **61**, 443–454 (2014).

22. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R. & Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016).

23. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061* (2016).

24. Qin, H., Gong, R., Liu, X., Bai, X., Song, J. & Sebe, N. Binary neural networks: A survey. *Pattern Recognit.* **105**, 107281 (2020).

25. Zhao, W., Ma, T., Gong, X., Zhang, B. & Doermann, D. A review of recent advances of binary neural networks for edge computing. *IEEE J. on Miniaturization for Air Space Syst.* **2**, 25–35 (2020).

26. Chang, Y.-F., O'Donnell, J. A., Acosta, T., Kotlyar, R., Chen, A., Quintero, P. A., Strutt, N., Golonzka, O., Connor, C. & Hicks, J. envm rram reliability performance and modeling in 22ffl finfet technology. In *2020 IEEE International Reliability Physics Symposium (IRPS)*, 1–4 (IEEE, 2020).

27. Ben Slimane, A., Michaud, A., Mauguin, O., Lafosse, X., Bercegol, A., Lombez, L., Harmand, J.-C. & Collin, S. 1.73 ev algaas/ingap heterojunction solar cell grown by mbe with 18.7% efficiency. *Prog. Photovoltaics: Res. Appl.* **28**, 393–402 (2020).

28. Van der Maaten, L. & Hinton, G. Visualizing data using t-sne. *J. machine learning research* **9** (2008).

29. Hirtzlin, T., Bocquet, M., Klein, J.-O., Nowak, E., Vianello, E., Portal, J.-M. & Querlioz, D. Outstanding bit error tolerance of resistive ram-based binarized neural networks. *arXiv preprint arXiv:1904.03652* (2019).

30. Buschjäger, S., Chen, J.-J., Chen, K.-H., Günzel, M., Hakert, C., Morik, K., Novkin, R., Pfahler, L. & Yayla, M. Margin-maximization in binarized neural networks for optimizing bit error tolerance. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 673–678 (IEEE, 2021).

31. Golonzka, O., Arslan, U., Bai, P., Bohr, M., Baykan, O., Chang, Y., Chaudhari, A., Chen, A., Clarke, J., Connor, C. *et al.* Non-volatile rram embedded into 22ffl finfet technology. In *2019 Symposium on VLSI Technology*, T230–T231 (IEEE, 2019).

32. Chen, H.-L., Cattoni, A., De Lépinau, R., Walker, A. W., Höhn, O., Lackner, D., Siefer, G., Faustini, M., Vandamme, N., Goffard, J. *et al.* A 19.9%-efficient ultrathin solar cell based on a 205-nm-thick gaas absorber and a silver nanostructured back mirror. *Nat. Energy* **4**, 761–767 (2019).

33. Massiot, I., Cattoni, A. & Collin, S. Progress and prospects for ultrathin solar cells. *Nat. Energy* **5**, 959–972 (2020).

34. Yoon, S., Carreon-Bautista, S. & Sánchez-Sinencio, E. An area efficient thermal energy harvester with reconfigurable capacitor charge pump for iot applications. *IEEE Transactions on Circuits Syst. II: Express Briefs* **65**, 1974–1978 (2018).

35. Loshchilov, I. & Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016).

Appendix C

Annexe 3 : In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling

nature electronics

ARTICLES

https://doi.org/10.1038/s41928-020-00523-3

Check for updates

In situ learning using intrinsic memristor variability via Markov chain Monte Carlo sampling

Thomas Dalgaty[1]✉, Niccolo Castellani[1], Clément Turck[2], Kamel-Eddine Harabi[2], Damien Querlioz[2]✉ and Elisa Vianello[1]✉

Resistive memory technologies could be used to create intelligent systems that learn locally at the edge. However, current approaches typically use learning algorithms that cannot be reconciled with the intrinsic non-idealities of resistive memory, particularly cycle-to-cycle variability. Here, we report a machine learning scheme that exploits memristor variability to implement Markov chain Monte Carlo sampling in a fabricated array of 16,384 devices configured as a Bayesian machine learning model. We apply the approach experimentally to carry out malignant tissue recognition and heart arrhythmia detection tasks, and, using a calibrated simulator, address the cartpole reinforcement learning task. Our approach demonstrates robustness to device degradation at ten million endurance cycles, and, based on circuit and system-level simulations, the total energy required to train the models is estimated to be on the order of microjoules, which is notably lower than in complementary metal–oxide–semiconductor (CMOS)-based approaches.

An exciting target for the future of computing is the development of a standalone system capable of learning, adapting and acting locally at the edge[1]—and thus independent of the cloud—while simultaneously working within constraints in terms of energy consumption, data availability and memory size. Currently, there is no commercial system that can meet these requirements, but edge learning is an application domain that is expected to grow over the coming decade[2]. However, the development of edge learning faces particular challenges given the noisy and limited raw sensory information encountered in the complex environments where such systems could be deployed. This could be, for example, an implanted medical system required to locally update its operation based on the evolving state of a patient.

Machine learning provides the enabling models and algorithms for these systems, but, until recently, little attention has been given to the hardware that underpins their computation. Machine learning models are trained using general-purpose hardware based on the von Neumann architecture[3]. This architecture has spatially separated processing and memory, which is not ideal for energy-efficient learning. For example, state-of-the-art performance in machine learning is currently being obtained with neural network models that feature a very high number of parameters[4], which are determined using a backpropagation algorithm[5] and a large volume of data. The energy required to train these models can be considerable due to the transfer of large quantities of information between the memory and processing centres of the hardware[6,7]. Although cloud computing platforms offer a centralized solution for such energy- and data-intensive training, these demands are not consistent with the requirements of edge learning[1,2]. For edge applications, it is likely that the von Neumann approach will need to be abandoned in favour of an alternative in which memory and processing coexist in the same location.

Resistive random access memory (RRAM) technologies, often referred to as memristors[8], are a promising approach for the development of in-memory computing systems due to their efficient implementation—relying simply on Kirchoff's current law—of the dot-product (or multiply-and-accumulate) operation used in machine learning[9] (Fig. 1a). RRAM comes in many forms[10–13], and intense effort is currently directed towards it use as synaptic elements in hardware-based artificial neural networks for edge computing systems[9,14–21]. Approaches for in situ training of these systems revolve around in-memory implementations of backpropagation algorithms[15,17–19]. Implementing these algorithms remains challenging because of multiple non-ideal device properties, such as nonlinear conductance modulation[22], a lack of stable multi-level conductance states[14,23] and device variability[24].

Although several non-ideality mitigation techniques[15–18,20,21] have been developed that enhance the accuracy of in situ training, these device properties lead to a performance that is lower than that obtained on conventional computing systems[25,26]. Approaches based on neuroscience-inspired learning algorithms, such as spike-timing-dependent plasticity, instead feature resilience and can sometimes benefit from device non-idealities[27–29]. However, these models cannot yet match state-of-the-art machine learning models when applied to practical tasks. Alternatively, it is possible to actively embrace resistive memory non-idealities. For example, the cycle-to-cycle conductance state variability, sometimes used as a source of entropy in random number generation[30], can also be exploited in stochastic artificial intelligence algorithms such as Bayesian reasoning[31,32], population coding neural networks[33] and in-memory optimization[34,35]. These approaches, however, sacrifice conductance non-volatility, which is the basis of resistive memory's potential for efficient in-memory computing.

In this Article, we report an approach that simultaneously exploits conductance variability and conductance non-volatility without requiring mitigation of other device non-idealities. We show that cycle-to-cycle conductance variability in resistive memories can be viewed as physical random variables that can be exploited to implement in-memory Markov chain Monte Carlo (MCMC) sampling algorithms[36]. In particular, we demonstrate how a resistive-memory-based Metropolis–Hastings MCMC sampling approach can be used to train, in situ, a Bayesian machine learning

[1]CEA, LETI, Université Grenoble Alpes, Grenoble, France. [2]Université Paris-Saclay, CNRS, Centre de Nanosciences et de Nanotechnologies, Palaiseau, France. ✉e-mail: thomas.dalgaty@cea.fr; damien.querlioz@c2n.upsaclay.fr; elisa.vianello@cea.fr

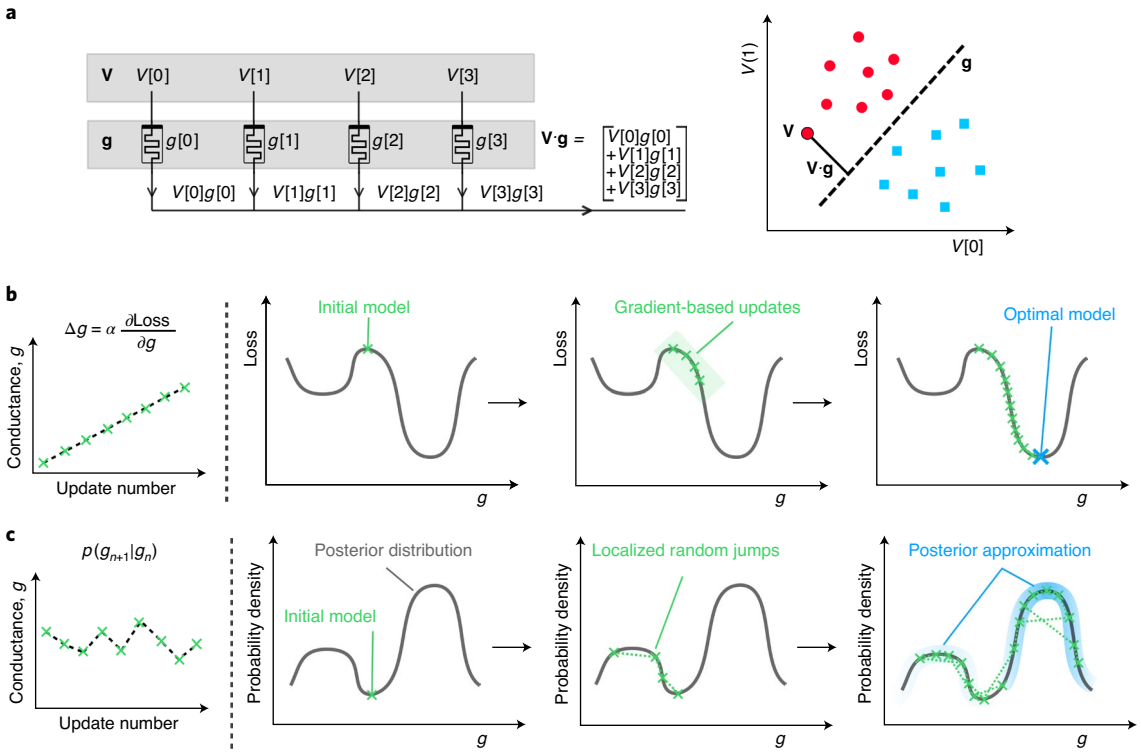


Fig. 1 | Strategies for training RRAM-based models. a, Left: a conductance model **g**, composed of four resistive memory elements, defines a linear boundary that (right) separates two classes of data (red circles from blue squares). Through application of a voltage vector **V** to the top electrode of the parallel resistive memories, the summed current flowing out of the common node at the bottom electrode is equivalent to the dot product $\mathbf{V} \cdot \mathbf{g}$, which can then be used to determine to what class the data point **V** belongs. **b**, Left: gradient-based learning algorithms iteratively compute the derivative of an error metric with respect to a conductance model **g**, multiplied by a learning rate α, to determine updates to be applied to the g parameters. The ideal RRAM device should be capable of high precision and linear conductance updates. Right: the three panels show the gradient-descent algorithm for an increasing number of model updates (green crosses). From an initial model, the algorithm performs gradient-based updates until it converges to a local minimum in error. **c**, Left: sampling algorithms use a proposal distribution $p(g_{n+1}|g_n)$ to propose random updates to model conductance parameters, which are then either accepted or rejected. The ideal RRAM device for sampling algorithms should offer random conductance updates deriving from a known probability distribution. Right: the three panels illustrate how a sampling algorithm performs local random jumps on the posterior distribution for an increasing number of sampling operations. From an initial model, the proposal distribution is used to generate a series of localized random jumps (dashed green lines), which are then either accepted (green crosses) or rejected. The algorithm tends to accept models of a higher probability density on the posterior distribution. After a sufficient number of iterations the accepted models can be used together as an approximation of the posterior distribution (blue shading).

model realized in an array of resistive memories. The devices that perform the critical sampling operations are also those that store the parameters of the Bayesian model in their non-volatile conductance states. This eliminates the need to transport information between processing and memory, and instead relies on the physical responses of nanoscale devices under application of voltage pulses inside the memory circuit itself.

To illustrate the practicality of RRAM-based MCMC sampling, we implement an experimental system that consists of a computer-in-the-loop with a fabricated array of 16,384 resistive memory devices in a one-transistor–one-resistor (1T1R) configuration, which are organized by the computer to realize a Bayesian machine learning model. We train the system to solve classification tasks including the detection of malignant breast tissue samples and the detection of arrhythmic heartbeats. We also apply the approach, using behavioural simulations calibrated on array-level variability characterization data, to the cartpole reinforcement learning task. Benchmarked against deterministic software-based neural network models, we find that the resulting Bayesian models perform better, and that RRAM-based MCMC trains a full model with orders of magnitude fewer programming operations compared to existing RRAM-based backpropagation approaches. Finally, through the design and simulation of a fully integrated implementation of our approach, we compare the training energy of our approach with that required using conventional complementary metal–oxide–semiconductor (CMOS) approaches, and observe an energy reduction of several orders of magnitude.

Resistive-memory-based MCMC sampling

Arrays of resistive memory devices are capable of efficient in-memory implementations of ex situ trained machine learning models[9,14]. Considering an RRAM-based logistic regression classifier, one of the most canonical models in machine learning, the circuit of M parallel devices shown in Fig. 1a defines a hyper-plane that can separate two classes of data. Each parameter of the logistic regression model is defined by the conductance of one of the M devices. The response of this conductance-based model, **g**, can be inferred by presenting a voltage vector **V**, encoding a data point, to the top terminals of the devices and sensing the current that flows out of the common, bottom node. This current is equivalent to the dot product between the two vectors, $\mathbf{V} \cdot \mathbf{g}$.

The dominant approaches for training such RRAM-based models in situ are gradient-based learning algorithms, whereby a loss metric is differentiated with respect to the current parameters of

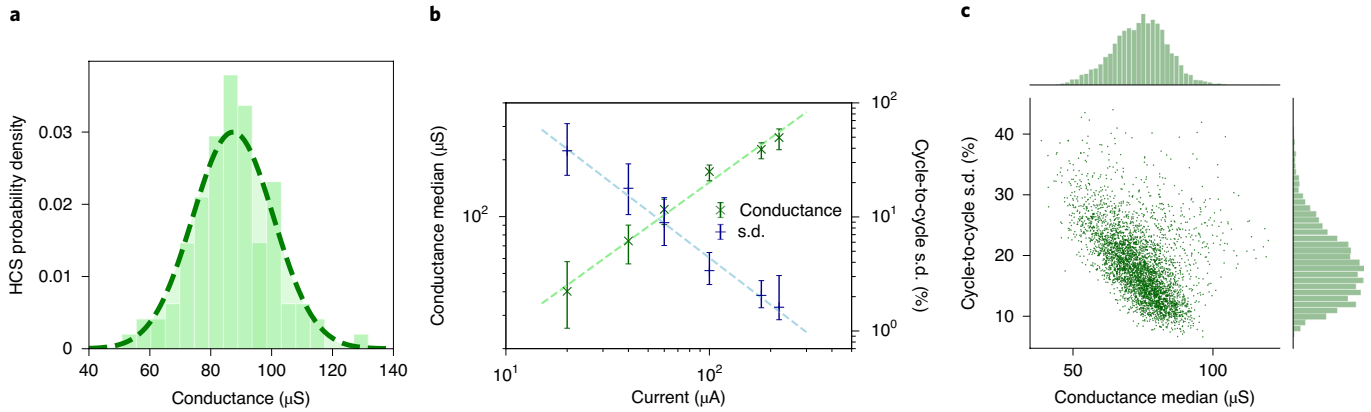


Fig. 2 | Electrical characterization of OxRAM cycle-to-cycle and device-to-device variability. a, Probability density of the HCS cycle-to-cycle variability for a single OxRAM device, measured over 100 RESET/SET cycles (Methods) and fitted with a normal distribution (dashed line). **b**, Cycle-to-cycle conductance median and standard deviation for a population of 4,096 devices, for a range of SET programming current (Methods). Both relationships are fit with a power law. The device-to-device variability in the quantities over the 4,096 devices, extending to the 95th and 5th percentiles (two standard deviations), is shown with error bars at each point. **c**, Joint distribution of the conductance median and standard deviation of each device in the population, where 4,096 devices have been RESET/SET-cycled 100 times under the same programming conditions (Methods), and the resulting median conductance and standard deviation of each device have been plotted as a single green point, illustrating the device-to-device variability within a population. Two histograms on opposing axes show the probability densities for the conductance median and standard deviation independently.

the model. The resulting derivative provides the conductance updates required to guide the model down the slope of a gradient until it settles into a minimum (Fig. 1b). However, performing this type of training in-memory is extremely challenging due to the nonlinear and random characteristics of resistive memory technologies, which do not naturally offer an adequate precision in these updates[14,22–24]. Furthermore, in such a deterministic modelling approach (Fig. 1a), each parameter is described by a single value, and it is not possible to account for parameter uncertainty. Capturing uncertainty in parameters is important when faced with the limited, possibly incomplete, data and noisy sensory readings encountered at the edge. To account for uncertainty, it is preferable to construct a Bayesian model[37,38]. In this case, parameters are represented, not by single values, but by probability distributions. The distribution of all of the parameters of a Bayesian model, given observed data, is called the posterior distribution. As an analogue to deriving an optimal deterministic model through gradient-based updates, the objective in Bayesian machine learning is to learn an approximation of the posterior distribution. For this purpose, sampling algorithms, most often MCMC sampling[36], can be employed. Instead of descending an error gradient, MCMC sampling makes localized random jumps on the posterior distribution (Fig. 1c). The algorithm jumps from a current location in the model space $\mathbf{g}$ to a proposed location $\mathbf{g}_p$, according to a proposal distribution $p(\mathbf{g}_p|\mathbf{g})$, usually a normal random variable. By comparing the proposed and current models, a decision is made on whether to accept or reject the proposed model. If accepted, the next random jump is made from this newly accepted model. After a sufficiently large number of iterations, the accepted samples describe, together, an approximation of the posterior distribution.

In this Article we realize that, in stark contrast to the case of gradient-based learning algorithms, the properties of resistive memories are incredibly well suited to the requirements of MCMC sampling algorithms. This is because the cycle-to-cycle conductance variability, inherent to device programming, is not a nuisance to be mitigated, but is instead a computational resource that can be leveraged by viewing resistive memory devices as physical random variables. More specifically, we exploit the variability of hafnium-dioxide-based random access memory[13] (OxRAM), co-integrated into a 130-nm CMOS process[39] (Methods). The conductivity of an OxRAM device can be modified through application of voltage waveforms, which, through reduction–oxidation reactions at an interfacial oxygen reservoir between the oxide and top electrode, create or rupture a conductive oxygen-vacancy filament between the electrodes. The device can be SET into a high conductive state (HCS) by applying a positive voltage to the top terminal of the device, while grounding the bottom, and thereafter RESET into a low conductive state (LCS) by applying a positive voltage to the bottom electrode while grounding the top.

Each time the device is SET, a unique HCS conductance is achieved, resulting from the random redistribution of oxygen vacancies within the oxide[24] on consecutive programming cycles. If the HCS conductance is measured over successive cycles, a normally distributed cycle-to-cycle conductance probability density emerges (Fig. 2a). The SET operation is therefore analogous to drawing a random sample from a normal distribution. In addition, the median conductance of this probability distribution can be controlled by limiting the SET programming current (I_{SET}) via the gate-source voltage of a series transistor. The relationship between the conductance median and SET programming current follows a power law[40], and the standard deviation of the distribution also depends on the SET programming current (Fig. 2b)—these quantities are also subject to device-to-device variations (Fig. 2c). Therefore, manifested in the physical response of these nanoscale devices, we find the essential computational ingredient required to implement in-memory MCMC sampling algorithms—a physical normal random variable that can be harnessed to propose new models based on the current one.

We propose that the $N \times M$ resistive memory array depicted in Fig. 3a can be trained through MCMC sampling and then store, in the distribution of its non-volatile conductance states, the resulting posterior distribution of a Bayesian model. A single deterministic model, $\mathbf{g}_n$, is stored in each of the rows, where its parameters are encoded by the conductance difference between positive $\mathbf{g}_{+n}$ and negative $\mathbf{g}_{-n}$ sets of devices—allowing for each parameter to be either positive or negative (Fig. 3b). The principle of our in situ learning approach is to generate at each row a proposed model, based on the model in the previous row, in line with the Metropolis–Hastings MCMC sampling algorithm[36] (see Methods for a detailed description). Each parameter of the proposed model can be

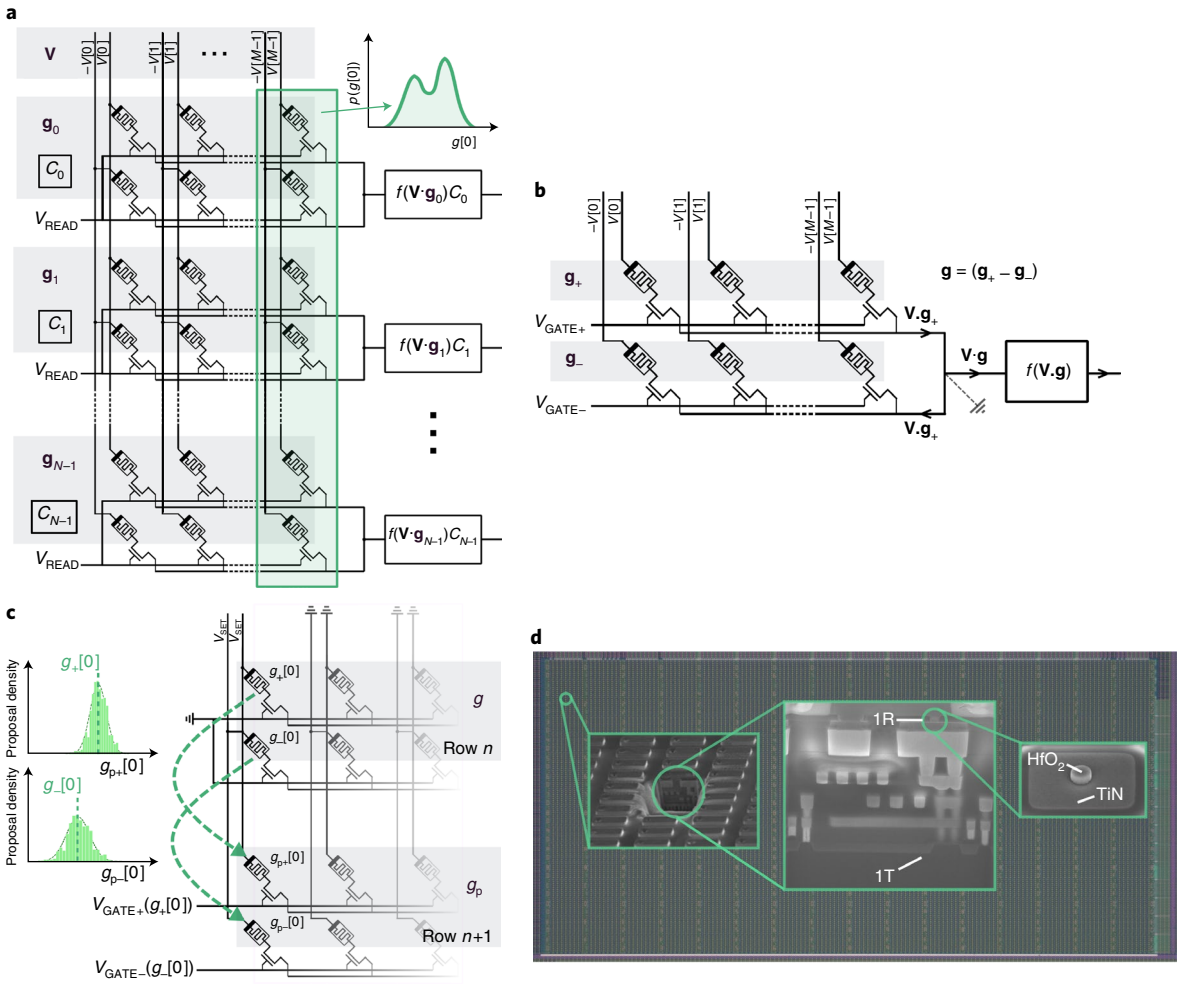


Fig. 3 | Implementation of Metropolis–Hastings MCMC sampling on a fabricated RRAM array. a, Memory array architecture where RRAM conductances store the posterior approximation. Each of the N rows stores a single conductance model $\mathbf{g}_n$, contains a digital counter element C_n and applies the function $f()C_n$ to the current flowing out of the row. **b**, A single array row is the differential conductance between conductance vectors $\mathbf{g}_+$ and $\mathbf{g}_-$. A positive voltage vector, $\mathbf{V}$, is applied to the top electrodes of $\mathbf{g}_+$, and an equivalent negative voltage vector, $-\mathbf{V}$, is applied over the top electrodes of $\mathbf{g}_-$. If the common, bottom node is pinned at a virtual ground (dashed ground symbol), the current flowing out of the row is equal to the dot product $\mathbf{V}\cdot\mathbf{g}$. **c**, Model proposal step between two array rows. Using the known relationship between SET programming current and the conductances read in row n (Methods), devices (pointed to by the dashed arrows) $g_{p+}[0]$ and $g_{p-}[0]$ in the $(n+1)$th array row are SET. The SET programming currents are proportional to the conductances of the corresponding devices in the nth row ($g_+[0]$ and $g_-[0]$). This thereby samples new conductance values for $g_{p+}[0]$ and $g_{p-}[0]$, in the $(n+1)$th row, from normal random variables with medians equal to the conductances of the devices $g_+[0]$ and $g_-[0]$ in the nth row (left, green distributions). The SET programming currents are determined by applying appropriate voltages V_{GATE+} and V_{GATE-} to the transistor gates of row $n+1$. **d**, The OxRAM array used in the experiments (Methods). An optical microscopy image of the fabricated array is shown in the background. Scanning electron microscopy images are superimposed on top. Left: a focused ion beam etch reveals the cross-section of a 1T1R structure (centre). In the front-end-of-line, a transistor (1T) acts as a selector for the OxRAM device (1R) integrated in the back-end-of-line. Right: imaged before deposition of the top electrode titanium layer, a 10-nm-thick, 300-nm-wide mesa of HfO_2 rests on a TiN bottom electrode.

generated naturally using the OxRAM physical random variable. This is achieved by performing a SET operation on each device in the row with a programming current that samples a new conductance value from a normal distribution, provided by its cycle-to-cycle variability, centred on that of the corresponding device in the previous row, as depicted in Fig. 3c. By computing a quantity called the acceptance ratio (Methods), a decision is made on whether this proposed model should be accepted or rejected. If rejected, the row is reprogrammed under the same conditions, thereby generating a new proposed model. Additionally, the value of a digital counter, C_n, which is associated with the previous row, is incremented by one. By tracking the number of rejections in this manner, the contribution of the model in each row to the overall probability density of the posterior approximation (Fig. 1c) is taken into account. If the proposed model is instead accepted, the process is repeated at the next row, and so on until the algorithm arrives at the final row of the array.

At this point, the distribution of programmed differential conductances in each of the array columns corresponds to the learned distributions of each of the Bayesian model parameters—the posterior distribution. After training, the learned posterior distribution in the array can be applied to a task through inference (Methods).

Supervised learning

We now apply in situ RRAM-based Metropolis–Hastings MCMC sampling to supervised learning tasks. This is achieved experimentally using a computer-in-the-loop with a fabricated array of 16,384 OxRAM 1T1R structures, as shown in Fig. 3d. In this 1T1R

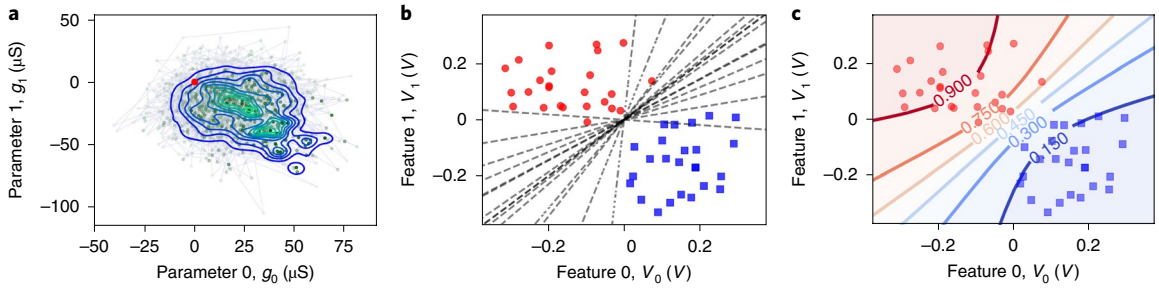


Fig. 4 | Experimental results on the illustrative two-dimensional dataset. a, Posterior distribution stored within the memory array after the training experiment. The two differential conductance parameters of each accepted model are plotted as points in the conductance plane (or model space). The initial model stored in the zeroth row is shown as a larger red dot. The models accepted into the subsequent array rows are plotted as green points with an opacity proportional to the associated row counter value. The transparent lines connecting green points show the jumps on the posterior between successive array rows. The resulting posterior distribution is superimposed in a contour plot whereby blue and green contours denote low and high probability densities, respectively. **b**, The two classes of data (red circles and blue squares) and the hyper-planes defined by a subset of 15 stored models from randomly selected rows of the memory array. **c**, The probabilistic boundary that is described by the posterior distribution stored within the resistive memory array. Each of the contour lines is annotated with the probability that any point lying on it belongs to the class of the red data. The bounded regions between contours are coloured from red to blue, where red denotes high probability that a point within that shaded region belongs to the red class, and blue the contrary.

configuration, the source and bit lines, contacting the top and bottom device terminals, respectively, run parallel to each other instead of orthogonally as in the proposed memory circuit of Fig. 3a. In our experiment, the dot product realized at each row (Fig. 3) is therefore evaluated on the computer. As a function of the acceptance ratio calculation, the computer then configures voltage waveforms, which iteratively read and program the devices in the array, thus implementing the MCMC sampling algorithm in-memory (see Methods and Supplementary Fig. 1).

First, as an illustrative example, we train a Bayesian logistic regression model, realized in a 2,048 × 2 array, to separate two classes of artificially generated data—the red circles and blue squares in Fig. 4b (Methods). After the algorithm terminates, the non-volatile conductance states of the devices in the array give rise to the multi-modal posterior approximation plotted in Fig. 4a. Two distinct peaks emerge, denoting regions of high probability density where many of the accepted models are tightly packed. A randomly selected subset of these accepted models are plotted as hyper-planes in the space of the data in Fig. 4b, each defining a unique linear boundary separating the two clouds of data points belonging to each class. Through combination of all the accepted models, therein using the posterior approximation that now exists in the array, the probabilistic boundary between the two classes in Fig. 4c emerges. Any previously unseen data point can hereafter be assigned a probability of belonging to the class of red circles as a function of where it falls on this probability contour.

We next apply the experimental system to the classification of histologically stained breast tissue as malignant or benign[41], using a Bayesian logistic model based on a 256 × 16 array. Following model training, the differential conductance parameters programmed into the resistive memory array are plotted in a heatmap in Fig. 5a (raw conductance and SET current distributions are shown in Supplementary Figs. 2 and 3). Probability distributions of two parameters from the resulting posterior are shown alongside. To visualize this learning process, the classification accuracy of the accepted model in each array row, in addition to its corresponding counter value, are plotted in Fig. 5b. From an initial conductance model, which achieves a poor accuracy, the algorithm quickly converges onto the posterior after ~32 rows (Methods and Supplementary Fig. 4). Strikingly, the accuracy does not saturate but rather increases sharply during the first 32 rows (the 'burn-in'), then oscillates between high and medium accuracy conductance models. This is an important property of MCMC sampling algorithms: sub-optimal models are also accepted, although less frequently, ensuring that the true form of posterior probability density is uncovered.

After the algorithm terminates, the accuracy achieved on the testing set was 97%. Notably, the combined accuracy of all of the models in the posterior approximation is greater than the accuracy of any of the single accepted deterministic models alone. This training process was repeated in 100 further experiments and the resulting accuracy distribution is reported in Fig. 5c, achieving a median accuracy of 96.3% and, on some iterations, classifying over 98% of test points correctly. This median accuracy could be sustained for array sizes down to 96 × 16 (Supplementary Fig. 5). To benchmark this result, deterministic software-based perceptron and single hidden-layer neural networks, using a number of synapses equal to the number of devices in our experiment, were trained using backpropagation on the same task. The resulting accuracy distributions of these benchmarks over the 100 training iterations are largely similar, obtaining a median accuracy of 95.8%, as plotted in Fig. 5c (see Supplementary Fig. 6 for deeper networks). Additionally, RRAM-based MCMC was found to require between one and four orders of magnitude fewer device programming operations than RRAM-based backpropagation implementations[17,21] of the benchmark models (Supplementary Figs. 7 and 8).

Alongside conductance variability, another drawback of resistive memory technologies is their rapid degradation when subjected to repeated programming operations. We therefore RESET/SET-cycled an RRAM array one million times and, after each decade of cycling, performed RRAM-based MCMC training. After only 100,000 cycles, the array is already unusable in a standard memory application (Supplementary Fig. 9). In spite of this, the accuracy of the models trained after each decade, plotted in Fig. 5d, remain comfortably within the bounds observed using the fresh memory array (Fig. 5c). In a further experiment, the error probabilities measured on the same technology for 10,000,000 cycles are artificially imposed during the experiment (Methods); again, the accuracy of the resulting model remains within this boundary.

To address a task more representative of learning at the edge, we also applied RRAM-based MCMC to train a multi-layer Bayesian neural network experimentally to detect heart arrhythmias from electrocardiogram recordings[42] (Methods and Fig. 5e). Bayesian neural networks are the probabilistic counterparts of deterministic ones, where probability distributions are used to represent the synaptic weights, allowing them to incorporate uncertainty and therefore

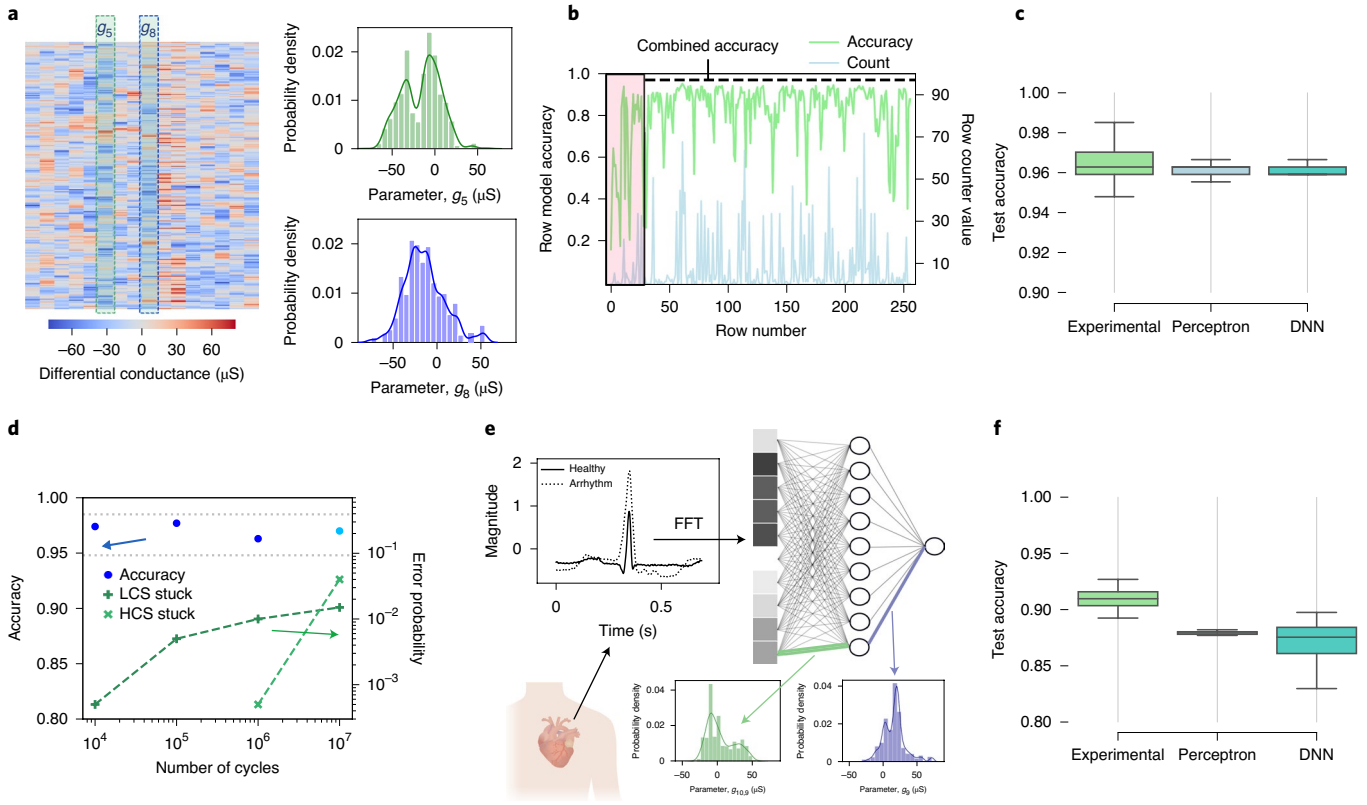


Fig. 5 | Experimental results on the supervised classification tasks. a, Left: heatmap of the differential conductance pairs in the 256 × 16 array after a training experiment. Cells within the heatmap are coloured from blue to red, indicating the sign and magnitude of each conductance parameter. The row counter weighted distributions within columns five and eight, corresponding to two learned model parameters, are plotted in green and blue histograms, respectively, and fitted with kernel density estimations. **b**, Accuracy, evaluated on the dataset, of the conductance model (green) and the row counter value (blue) for each of the 256 rows. The first 32 rows, contained within a red-shaded rectangle, have been accepted during the burn-in period. Discarding these rows, the combined accuracy of the remaining rows on the test dataset is 97%, indicated by the horizontal dashed line. **c**, Boxplots showing the test accuracy distributions over 100 separate train/test iterations for the malignant tissue recognition task using the same train/test split for the experimental set-up (left, green) and software-based neural network benchmark models (centre and right, blue; labelled perceptron and DNN). The coloured boxes span the upper and lower quartiles of accuracy, while the upper and lower whiskers extend to the maximum and minimum accuracies obtained over the 100 iterations. The median accuracy is indicated with a solid horizontal line. **d**, The experimental accuracy of the trained model after an increasing number of decades of endurance cycles. The permanent write error probabilities of devices stuck in the LCS and stuck in the HCS are also plotted on a secondary axis. Arrows have been annotated to indicate which curves correspond to which axes and grey horizontal dashed lines denote the lower and upper accuracies obtained over 100 training iterations using a fresh array. **e**, Depiction of the arrhythmic heartbeat detection experiment. Electrocardiograms (top left) recorded from the heart, potentially using an implanted cardioverter defibrillator[42] (bottom left), are used to extract 10 features through a fast Fourier transform (labelled as FFT), which are then used as the input for a multi-layer Bayesian neural network (top right). The Bayesian neural network is trained to detect arrhythmic beats. Two of the resulting synaptic weight distributions after training are also plotted (bottom right). **f**, Boxplots showing the test accuracy distributions over 100 separate train/test iterations of the arrhythmic heartbeat detection task using the same train/test split for the experimental set-up (left, green) and software-based neural network benchmark models (centre and right, blue; labelled perceptron and DNN). The coloured boxes span the upper and lower quartiles of accuracy, while the upper and lower whiskers extend to the maximum and minimum accuracies obtained over the 100 iterations. The median accuracy is indicated with a solid horizontal line. Body and heart in **e** adapted from InjuryMap. Distributed under a Creative Commons licence CC BY 4.0.

be robust to overfitting[38]. The trained model was tasked with the detection of arrhythmic beats in a previously unseen subject. The accuracy distribution over 100 training iterations is plotted in Fig. 5f, and the model obtains a median test accuracy of 91%, higher than that obtained by two software benchmark models. This highlights that, beyond being RRAM-compatible, Bayesian machine learning offers an alternative modelling method that appears well suited to the characteristics of edge learning tasks. The comparison between the total number of programming operations observed in our experiment and that required by RRAM-based backpropagation approaches applied to the benchmarks was repeated and, once again, our experimental approach achieved a very favourable programming efficiency (Supplementary Figs. 7 and 8).

Calibrated on an array-level variability characterization, a behavioural simulation of the experiment was developed (Methods) that accurately reproduces the experimental results (Supplementary Fig. 10). Using this simulator, we show in Supplementary Fig. 11 that our scheme is robust to device variability levels with a large deviation from measured values, suggesting its applicability to a broad range of technologies. The simulator was also used to demonstrate the extension of the approach to multi-class tasks using a Bayesian perceptron (Supplementary Fig. 12).

Reinforcement learning

We now demonstrate that the approach can be extended to reinforcement learning using the behavioural simulator. In contrast

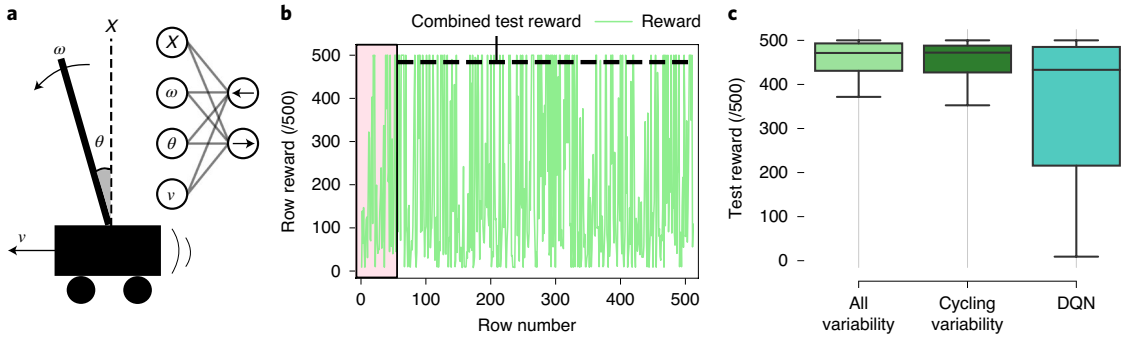


Fig. 6 | Behavioural simulation results on the cartpole reinforcement learning task. a, Diagram of the cartpole task, in which an agent learns how to accelerate left or right to maintain a pole balanced on top of a cart within 15° of normal (the vertical dashed line). The environment is described by four features: X, the x-position of the cart; θ, the angle of the pole to vertical; ω, the angular velocity at the tip of the pole; ν, the velocity of the cart. These four features serve as input to the perceptron model (upper right), where the two output neurons determine whether the agent accelerates to the left or right. **b**, Reward obtained during a training episode when each of the conductance models was accepted into one of the 512 array rows (the maximum reward is 500). A burn-in period of 64 rows is denoted with the red-shaded rectangle. The mean reward obtained over 100 test episodes using the combination of the models accepted after the burn-in, equal to 484 out of a maximum score of 500, is denoted with a horizontal dashed black line. **c**, Boxplots showing the distribution of the mean test reward obtained during 100 testing episodes achieved in the cartpole task over 100 separate train/test iterations. Behavioural simulation considering both device-to-device and cycle-to-cycle variability (left, light green), behavioural simulation considering only cycle-to-cycle variability (centre, dark green) and a DQN benchmark model (right, blue). The coloured boxes span the upper and lower quartiles of mean reward, the upper and lower whiskers extend to the maximum and minimum mean rewards obtained, and the median mean reward is indicated with a solid horizontal line.

to supervised learning, reinforcement learning does not require a labelled dataset—an appealing prospect for edge learning systems. Instead, a model is tasked with determining the actions of an agent based on interaction with its environment[43]. The agent observes, at each time step, information from the environment (**V**) and also takes an action (**a**). Resulting from the actions taken by the agent, a scalar reward (r) is received. The objective in reinforcement learning is to realize a model that allows an agent to take actions that maximize future reward. Here, we apply RRAM-based MCMC to learn a posterior distribution in terms or reward[44].

We address the cartpole control task[45] (Supplementary Videos 1 and 2), an important validation task in RRAM-based reinforcement learning[20,46], in which an agent learns how to control a pole balanced on top of a cart by accelerating to the left or right, as a function of four observed environmental variables (Fig. 6a). To achieve this, we employ a two-neuron Bayesian perceptron, effectively two 512 × 4 memory arrays. The output neuron with the largest response, as a function of the input features, dictates the action taken by the agent at each time step.

The training process is visualized by plotting the reward received when each row was accepted in Fig. 6b (see Supplementary Fig. 13 for row counts). After training, the agent then uses the learned posterior approximation to select actions over 100 testing episodes, achieving a mean test reward of 484 out of 500 (Methods). This procedure was repeated 100 times and the distribution of mean test reward is plotted in Fig. 6c. To benchmark this result, a software-based Deep-Q network[47] (DQN) was applied to the same task (Methods), with a number of synapses equal to the number of devices in our simulation. The median mean test reward of the DQN was only 420 and exhibited greater variability between training iterations than our model (Fig. 6c). Supplementary Fig. 6 shows a comparison with deeper models.

To assess the impact of device-to-device variability (Fig. 2c) on this task, the simulation was repeated without its consideration (Methods). The resulting test reward distribution (Fig. 6c) is seen to be largely unaffected, consistent with the supervised learning results (Supplementary Fig. 11). This result challenges the long-standing conception that device-to-device variability (Fig. 2c) is a disadvantage in RRAM-based machine learning that requires mitigation. However, this should not come as a surprise—unlike gradient-based optimization, where device-to-device variations impede the descent down a gradient, MCMC sampling actively leverages such randomness as a means of exploring the posterior.

Finally, to evaluate the energy efficiency of a fully integrated RRAM-based MCMC sampling system, we performed the design and simulation of the different CMOS elements that compose it, using standard analogue and digital integrated circuit design tools. The methodology and results are detailed in Supplementary Note 1, Supplementary Fig. 14 and Supplementary Table 1. To fully train the cartpole model, only 6.9 µJ would be required in the 130-nm technology used to design the test chip used throughout this Article (Fig. 3d). In a scaled 28-nm technology, this reduces to 3.6 µJ. By comparison, an optimized implementation of the MCMC sampling algorithm on a modern workstation processor consumes on the order of 600 mJ when applied to the same task. Projecting these results to the supervised learning experiments, it is estimated that 1.3 µJ and 4.7 µJ of energy, respectively, would be required to train the presented Bayesian logistic regression and neural network models in the 28-nm node.

Conclusion

We have demonstrated the potential of simultaneously harnessing resistive memories as physical random variables and as efficient dot-product engines via the development of in situ MCMC sampling algorithms. Our approach, which actively exploits cycle-to-cycle conductance variability, is resilient to device-to-device variability and extensive device aging (beyond the point where devices could be used as traditional memories). We have illustrated the potential of this approach in both supervised and reinforcement learning situations.

The appeal of RRAM-based MCMC is highlighted by two sets of our results. First, a reduced number of programming operations are required to train a full model with respect to RRAM-based backpropagation approaches, and there is a reduction in the energy of a full RRAM-based MCMC system relative to conventional CMOS solutions (Supplementary Note 1). Second, when faced with data representative of the edge (noisy and multi-modal electrocardiograms in the arrhythmia detection task), the robustness of Bayesian

machine learning to overfitting by incorporating uncertainty into learned parameters[37,38] was found to be crucial in outperforming the software-based deterministic benchmark neural networks trained through backpropagation; this illustrates that, beyond being compatible with the fundamental properties of resistive memory devices, RRAM-based Bayesian models are also particularly well suited to edge applications.

Our system could be used as the foundation (Supplementary Fig. 14 and Supplementary Note 1) for the design and fabrication of a standalone and fully integrated RRAM-based MCMC sampling chip, for applications outside the laboratory. One notable use case of RRAM-based MCMC, which builds on the promise of our arrhythmia detection results, is its application to energy-constrained learning and adaptation in implantable medical systems—a potentially revolutionary domain that is currently out of reach with existing commercial approaches[2]. This prospect is also supported by the fact that Bayesian network topologies are already employed in some biomedical machine learning applications[48].

Finally, the emerging class of 'scalable' MCMC algorithms[49–51], which offer a means of extending RRAM-based MCMC to larger models and datasets, should be explored in future systems.

Methods

HfO_2-based resistive-memory arrays and experimental set-up. Two versions of fabricated OxRAM memory arrays are used in the presentation of this Article. The first is a 4,096 (4k) device array (16 × 256 devices) of 1T1R structures. The second chip is a 16,384 (16k) device array (128 × 128 devices) of 1T1R structures. In each array, the OxRAM cell consists of a HfO_2 thin film sandwiched in a TiN/HfO_2/Ti/TiN stack. The HfO_2 and Ti layers are 5 nm thick and have a mesa structure that is 300 nm in diameter. The OxRAM stack is integrated into the back-end-of-line of a commercial 130-nm CMOS process. In the 4k device array, the n-type selector transistors are 6.7 µm wide. In the 16k array, the n-type selector transistors are 650 nm wide. Voltage pulses, generated off chip, can be applied across specific source (SL), bit (BL) and word lines (WL) which contact the OxRAM top electrodes, transistor sources and transistor gates, respectively. External control signals determine to what complement of SL, BL and WL the voltage pulses are applied by interfacing with CMOS circuits integrated with the arrays. Signals for the 4k device array are generated using the Keysight B1530 module and those for the 16k device array are generated by the RIFLE NplusT engineering test system. The RIFLE NplusT system can also run C++ programs that allow the system to act as a computer-in-the-loop with the 16k device array.

Before either chip can be used, it is necessary to form all the devices in the array. In the forming process, oxygen vacancies are introduced into the HfO_2 thin film through a voltage-induced dielectric breakdown. This is achieved by selecting devices in the array one at a time, in raster-scan fashion, and applying a voltage between the source and bit lines. At the same time, the current is limited to the order of microamperes by simultaneously applying an appropriate V_{WL} (transistor gate) voltage. A forming operation consists of the following conditions: 4k device array, $V_{SL}=4$ V, $V_{BL}=0$ V, $V_{WL}=0.85$ V; 16k device array, $V_{SL}=4$ V, $V_{BL}=0$ V, $V_{WL}=1.3$ V. After the devices have been formed, they are conditioned by cycling each device in the array between the LCS and the HCS 100 times. Unless otherwise specified, the standard RESET conditions used in the paper were as follows: 4k device array, $V_{SL}=0$ V, $V_{BL}=2.5$ V, $V_{WL}=3$ V; 16k device array, $V_{SL}=0$ V, $V_{BL}=2.6$ V, $V_{WL}=4.0$ V. Unless otherwise specified, the standard SET conditions used were as follows: 4k device array, $V_{SL}=2$ V, $V_{BL}=0$ V, $V_{WL}=1.2$ V; 16k device array, $V_{SL}=2.0$ V, $V_{BL}=0$ V, $V_{WL}=1.6$ V. The device conductances are determined by measuring the voltage drop over a known low-side shunt resistance connected in series with the selected SL in a read operation. Devices are read according to the following conditions: 4k device array, $V_{SL}=0.1$ V, $V_{BL}=0$ V, $V_{WL}=4.8$ V; 16k device array, $V_{SL}=0.4$ V, $V_{BL}=0$ V, $V_{WL}=4.0$ V. All off-chip generated voltage pulses for programming and reading have a pulse width of 1 µs.

Measurement of OxRAM HCS random variable properties. To characterize the properties of the HfO_2 physical random variable, as plotted in Fig. 2a–c, the 4k device array chip was used. This allows use of a larger selector transistor, which offers a greater range of SET programming currents. To measure the data plotted in Fig. 2b, a 4k device array was formed, conditioned and then RESET/SET-cycled 100 times over a range of nine word-line voltages (V_{WL}), corresponding to the range of SET programming currents in Fig. 2b. Each device was read after each SET operation. Between each step in V_{WL}, the devices were additionally RESET/SET-cycled 100 times under standard programming conditions ensuring that, for each of the 100 cycles at different V_{WL}, the initial conditions were the same. For the data plotted in Fig. 2a,c, a single device and all 4k devices in the 4k device array were respectively RESET/SET-cycled 500 times. The conductance was read after each SET operation. This conductance data were then processed and plotted using the Python libraries NumPy, SciPy, Seaborn and Matplotlib.

MCMC sampling experiments on the 16k device array. The computer-in-the-loop experiments (used to obtain the results in Figs. 4a–c and 5a–c) made use of the 16k device array interfaced to the C++ programmable RIFLE NplusT system (Supplementary Fig. 1). The devices in the array, which physically exist as a 128 × 128 array of 1T1R structures, were remapped into a virtual address space that realizes the structure presented in Fig. 3a. This was achieved by allocating pairs of sequential banks of M devices (for an M-parameter model) corresponding to the $\mathbf{g}_+$ and $\mathbf{g}_-$ conductance vectors to each of the N rows in Fig. 3a. A dot product was performed by reading the conductances of the devices composing $\mathbf{g}_+$ and $\mathbf{g}_-$ and subtracting them in the computer-in-the-loop to arrive at $\mathbf{g}$ and then performing the dot product between $\mathbf{V}$ and $\mathbf{g}$. Note that, in a future version of the system, by integrating appropriate circuits within the device array the dot product could be performed by simply applying the data points as read voltages and reading the output current as described in this Article.

Resistive-memory-based MCMC sampling begins by performing a RESET operation on each device in the array, rendering all devices in the LCS. Using the variable n to point to the row containing the current model, the algorithm begins with $n=0$. The devices in row n are SET. Because we do not have strong a priori belief on the initial model parameters, each device is SET using the lowest available V_{WL} (in this case 1.4 V), which corresponds to the lowest SET programming current (40 µA) (for a comparison of initialization strategies see Supplementary Fig. 15). As a result the standard deviation of the initial samples will be high (Fig. 2b), thereby capturing this uncertainty. The devices in the following row, $n+1$, are then programmed with SET programming currents proportional to the conductances read from the corresponding devices in row n. This results in a proposed model, $\mathbf{g}_p$, being generated in row $n+1$, in line with the proposal distribution offered by the cycle-to-cycle HCS conductance variability:

$$p\left(\mathbf{g}_p|\mathbf{g}\right)=\mathcal{N}(\mathbf{I}_{SET}(\mathbf{g}),\sigma(\mathbf{g})) \tag{1}$$

In doing this, a new conductance value is sampled for each device in row $n+1$ from a normal random variable with a median value corresponding to the same device in row n, offset by device-to-device variability (Fig. 2c) that is introduced when moving between successive rows.

The SET programming current in the proposal step is determined by the value of V_{WL} used to program each device in row $n+1$. To achieve this, a single look-up table is determined in an initial sweep step whereby the entire 16k device array is RESET/SET-cycled once per V_{WL} value that will be used in the experiment. For each of these values of V_{WL}, the median conductance read across the 16k device array is calculated and inserted in the corresponding entry in the table. Therefore, when programming, a device in the row containing the proposed model is required to read the conductance of the corresponding device in row n, and use the value of V_{WL} with the closest corresponding conductance as the V_{WL} used in the SET operation. In the experiments, the look-up table extended from 1.4 V to 1.8 V in discrete 20 mV steps, corresponding to SET programming currents in the range of 40–100 µA, and permitted median conductances in the range of 40–80 µS to be used (Supplementary Fig. 16). Programming the devices in row $n+1$ implements the model proposal step depicted in Fig. 3c.

In Metropolis–Hastings MCMC sampling, after proposing a new model, it is necessary to make a decision on whether to accept and record the proposed model $\mathbf{g}_p$ or reject and record, once again, the current model $\mathbf{g}$. This decision is made based on the calculation of a quantity named the acceptance ratio a. Because the proposal density is normally distributed (equation (1)), and therefore symmetrical, it can be written as

$$a=\frac{p(\mathbf{g}_p)}{p(\mathbf{g})}\frac{p(\mathbf{t}|\mathbf{g}_p,\mathbf{V})}{p(\mathbf{t}|\mathbf{g},\mathbf{V})} \tag{2}$$

This acceptance ratio is a number proportional to the product of the likelihood of a proposed model ($p(\mathbf{t}|\mathbf{g}_p,\mathbf{V})$) and a prior on the proposed model ($p(\mathbf{g}_p)$), divided by the product of the likelihood and prior of the current model. Given a dataset of D data points where A data points belong to the class the model is required to recognize ($t=1$) and B other data points to the class that the model should not recognize ($t=0$), the Bernoulli likelihood of a model is given by

$$p(\mathbf{t}|\mathbf{g},\mathbf{V})=\prod_{a=0}^{A}f(\mathbf{V}_{a,t=1}\cdot\mathbf{g})\times\prod_{b=0}^{B}\left(1-f(\mathbf{V}_{b,t=0}\cdot\mathbf{g})\right) \tag{3}$$

The function $f(\mathbf{V}\cdot\mathbf{g})$ depends on the specific formulation of the model. The prior of a model is given by

$$p(\mathbf{g})=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(\mathbf{g}-\mu)^2}{2\sigma^2}\right) \tag{4}$$

where the constant σ corresponds to the prior belief that the posterior distribution is a multi-dimensional normal distribution with a standard deviation of σ in each dimension. In all examples in this Article, the value of μ was set to zero.

These quantities are calculated on the computer-in-the-loop in logarithmic scale, constituting a summation over D points. Because MCMC sampling methods work with small datasets, where overfitting can be avoided through the incorporation of uncertainty into parameter estimations, it would be possible to subsample the original D data points if the dataset were unreasonably large. To decide if the proposed model should be accepted or rejected, a is compared to a uniform random number between 0 and 1 (u), generated on the computer using the C++ standard random math package. If a is less than u, then $\mathbf{g}_p$ is rejected. This is achieved by programming the devices in row $n+1$ back into the LCS and incrementing the counter at row n (that is, C_n) by one. The counter is a variable in C++ on the computer, although, as is the case for all functionality of the computer-in-the-loop, this will be integrated as a circuit on future implementations of the system. The devices in row $n+1$ are then SET once more under the same programming conditions, generating a new proposed model $\mathbf{g}_p$ at row $n+1$. This process repeats until a is found to be greater than u. When this is the case, $\mathbf{g}_p$ is accepted, whereby the counter at row $n+1$ is incremented by one and the model at row $n+1$ becomes the new current model ($n = n+1$). This new current model $\mathbf{g}$ is then used to propose a model, $\mathbf{g}_p$, at the next row in the array. The model stored at row $n-1$ is then left preserved in the non-volatile conductance states of the resistive memory devices, weighted by the counter value C_{n-1}. As this process repeats, and progresses down the rows of the memory array, the algorithm randomly walks around the posterior distribution, leaving information on its probability density imprinted into the non-volatile conductance states of the OxRAM devices and the row counter values. On arriving at the final row of the array, $n = N-1$, the training process terminates, resulting in a physical array of resistive memory devices that contains an approximation of the posterior distribution that can then be used in inference.

Performing inference consists of computing the dot product between a new, previously unseen data point ($\mathbf{V}_{new}$) and the model recorded in each array row. The response of each row is then multiplied by the value in each row counter. The summed response of each row in the array is then divided by the sum of all row counter values. This results in a scalar value that has been inferred from the N weighted samples from the posterior distribution approximation:

$$P(T_{new} = 1|\mathbf{V}, \mathbf{t}) = \frac{1}{\mathrm{Tot}} \sum_{n=\beta}^{N-1} C_n f(\mathbf{V}_{new} \cdot \mathbf{g}_n) \quad (5)$$

The summation considers only rows of an index greater than β, which determines the number of rows discarded to account for the burn-in period. The variable Tot is the sum of all row counter values recorded after the burn-in period.

Supervised learning experiments. The memory array used in the supervised learning experiments is configured as a Bayesian logistic regression model by using the row function block:

$$f(\mathbf{V} \cdot \mathbf{g}) = \frac{1}{1 + e^{-S(\mathbf{V} \cdot \mathbf{g})}} \quad (6)$$

where S is a scaling parameter. This logistic function limits the response of the row dot product into a probability between 0 and 1. When configured as such, the likelihood of a conductance model is calculated as

$$p(\mathbf{t}|\mathbf{g}, \mathbf{V}) = \prod_{a=0}^{A} \left(\frac{1}{1 + e^{-S(\mathbf{V}_{a,t=1} \cdot \mathbf{g})}} \right) \times \prod_{b=0}^{B} \left(1 - \frac{1}{1 + e^{-S(\mathbf{V}_{b,t=0} \cdot \mathbf{g})}} \right) \quad (7)$$

After training, inference can performed on a new data point $\mathbf{V}_{new}$, whereby it is assigned a probability of belonging to the class $t = 1$ by computing

$$P(T_{new} = 1|\mathbf{V}, \mathbf{t}) = \frac{1}{\mathrm{Tot}} \sum_{n=\beta}^{N-1} \frac{C_n}{1 + e^{-S(\mathbf{V}_{new} \cdot \mathbf{g}_n)}} \quad (8)$$

In the first supervised learning task, the random sampling library from NumPy was used to generate 50 samples from a two-dimensional normal distribution centred at the origin. Half of the points were assigned a class label of $t = 1$ and shifted up and to the left, while the other half were shifted down and to the right (by the same value) and labelled $t = 0$, thus providing an artificial linearly separable dataset of two classes for an illustrative demonstration of the system. For the second supervised learning task, the Wisconsin breast cancer dataset was used, which consists of 569 data points with class labels 'malignant' ($t = 1$) or 'benign' ($t = 0$)[41]. The dataset was accessed through the Scikit-learn library and the random shuffle function from NumPy was used to sort the dataset into 369 training points and 200 test points, which were used in each of the 100 train/test iterations. By implementation of the Chi2 feature selection algorithm[52], available through Scikit-learn, the number of features was reduced to 16, as this corresponded to the number of array columns used in the experiment. A further data pre-processing step was performed using the scale function from Scikit-learn to centre the dataset around the origin such that the model did not require an additional bias parameter. If this step was not performed, an extra column could be added to the array, which would then learn the distribution of the bias parameter. During training, the algorithm was configured to recognize data points corresponding to malignant tissue samples ($t = 1$). During the inference time, the 200 previously unseen data points from the test split were assigned a probability of being malignant using equation (8). Output probabilities greater than or equal to 0.5 corresponded to a prediction of the sample being malignant ($t = 1$), and probabilities of less than 0.5 corresponded to a prediction of the sample being benign ($t = 0$). The reported test accuracy corresponds to the fraction of the 200 test data points that were correctly classified.

To force permanent write failures for the study impact of endurance cycling on RRAM-based MCMC sampling, C++ code was written whereby, after reading a new set of conductances from the memory array, the read bits were set with a probability to be either a stuck-in-LCS or a stuck-in-HCS value, according to the literature[39]. When determining what voltage to apply to the gates of the 1T1R structures in the subsequent row, the lowest available gate voltage was applied to a stuck-in-LCS bit and the maximum available gate voltage was applied for a stuck-in-HCS bit.

When performing training iterations after the successive decades of endurance cycling as plotted in Fig. 5d, the iteration of RRAM-based MCMC sampling was performed as described, without recalibration of the look-up table linking the read conductance value to the gate voltage applied to the 1T1R structure of the corresponding device in the subsequent row. In addition, because it was determined that arrays sizes in the range of 96 × 16 to 256 × 16 were able to obtain the same performance (Supplementary Fig. 5), a 4,096-device subset of the 16,384-device array was subject to endurance cycling, allowing a fourfold reduction in the time required to perform the endurance experiment to the order of days. As such, the results obtained in Fig. 5d were obtained with a 128 × 16 array.

For the arrhythmic heartbeat detection task, we downloaded data from the MIT-BIH heart arrhythmia database[42], which comprised half-hour dual-channel electrocardiogram recordings from 48 subjects. Single heartbeats were isolated from each recording into a 700-ms time series and centred on the R-wave peak of the beat. The FFT function from NumPy was then used to generate a frequency spectrum of each time series and the lowest five frequency components from each channel were used as the 10 features used to describe each heartbeat as a static data point. Each heartbeat was labelled as either a normal, healthy heartbeat or as a heartbeat exhibiting signs of arrhythmia. A subset of 250 data points were taken randomly from 47 of the subjects and then used to train the models. Then, to test the models, all of the data points from one, previously unseen, subject were used. The RRAM-based Bayesian neural network model, with a hidden-layer of nine sigmoid neurons, was trained in the same fashion as the RRAM-based logistic regression model. Because each sample requires 198 devices, and given the size of the experimental array, there was capacity for storing only 82 models in our experiment. To account for the burn-in period, the first 32 sampled models were discarded.

MCMC sampling behavioural simulator. A custom behavioural simulation of our experiment was developed in Python, implementing the presented Metropolis–Hastings MCMC sampling algorithm. The proposal distributions were calibrated on data measured on the 4k device array. A normal distribution, using the random function suite from the NumPy library, was used to sample proposed models ($\mathbf{g}_p$) from current models ($\mathbf{g}$). The standard deviation (s.d.) of the sample was determined based on the data plotted in Fig. 2b, where the median relationship was seen to follow the power law

$$\text{s.d.} = a \times {I_{SET}}^b \quad (9)$$

with constants $a = 0.093A^{1-b}$ and $b = 0.48$. The current I_{SET} is determined based on the conductances from the current model using the data in Fig. 2b, where

$$I_{SET} = \left(\frac{g}{d}\right)^{1/c} \quad (10)$$

with constants $c = 0.78$ and $d = 0.19S/A^c$ (plotted in Fig. 2b). To incorporate device-to-device variability, the s.d. in c, fitted for individual devices in the 4k device array (Supplementary Fig. 17), was found to be equal to $e = 0.096S/A^c$. The likelihood and acceptance ratio calculations were performed in the log domain.

Reinforcement learning simulation. The Python library gym was used to simulate the cartpole environment. The cartpole environment provides four features to the behavioural simulator at each time step of the simulation. The behavioural simulator then specifies the actions to be taken by the agent in the environment at the next simulation time step. Two simulated 512 × 4 arrays were used. One array encoded the accelerate left action, while the other encoded the accelerate right action. The two arrays compete in a winner-take-all fashion to determine the actions taken by the agent. During training, the output of the array rows containing the two proposed models is used to determine the actions of the agent at each time step. Under application of a new observation from the environment $\mathbf{V}$, the response

$$f(\mathbf{V} \cdot \mathbf{g}) = S\ \mathbf{V} \cdot \mathbf{g} \quad (11)$$

is calculated, where S is a scalar constant. The two arrays are treated as if they are a single model. Rows of equivalent index in both arrays share a common row counter and are programmed and evaluated at the same time. The devices within

the zeroth row of both arrays are initially SET by sampling from a normal random variable with the lowest available conductance median (in this simulation, the conductance range extended from 50 μS to 200 μS). The initial model is evaluated during a training episode where the cumulative reward received during the episode is recorded. For each time step that the agent does not allow the pole to rotate 15° away from the perpendicular or does not move outside the bounds of the environment (both resulting in early termination of the episode), the agent receives a +1 reward. If the agent maintains the pole balanced for 500 time steps, the episode terminates, resulting in an episode in which the agent has received the maximum possible score of 500. Respective models are then generated at the first row of each array by sampling new models from normal random variables with medians equal to the conductances of the corresponding devices in the zeroth row, offset by device-to-device variability. The two proposed models determine the actions of the agent during the following training episode, and the agent accelerates to either the left or the right at each time step of the episode as a function of which array exhibited the greater response (equation (11)). As a function of the cumulative reward achieved during this training episode, a decision is made on whether to accept or reject the proposed model using the acceptance ratio

$$a = \frac{p(\mathbf{g}_p)}{p(\mathbf{g})}\frac{p(r|\mathbf{g}_p, \mathbf{V}, \mathbf{a})}{p(r|\mathbf{g}, \mathbf{V}, \mathbf{a})}\kappa^{-1} \quad (12)$$

Instead of using the ratio of likelihoods of the proposed and current models as in the supervised case, the ratio of episodic rewards for a model $\mathbf{g}$, episodic observations $\mathbf{V}$ and episodic actions $\mathbf{a}$ is used: this is simply the scalar reward value obtained after an episode acting according to a proposed model $\mathbf{g}_p$, divided by the reward received when the current model $\mathbf{g}$ is accepted. This demonstrates a particular strength of applying MCMC sampling in a reinforcement learning setting, relative to supervised learning, because the calculation of the acceptance ratio can be achieved in a single calculation instead of an operation involving a sum over all data points in the log domain. The prior calculation is the same as in equation (4). The acceptance ratio is then multiplied by a constant κ^{-1}, which acts as a hyper-parameter that determines the extent of exploration from higher to lower reward regions on the posterior. If the acceptance ratio is less than a uniform random number generated between 0 and 1, then the proposed model at the first row is rejected and the row counter, C_0, is incremented by one. A new model is then proposed at the first row. If this new model achieves a cumulative reward during the next training episode, ($p(r|\mathbf{g}_p, \mathbf{V}, \mathbf{a})$), such that the acceptance ratio is now greater than a uniform random number, the two models within the first rows of both arrays are accepted and then become the current models. The reward that was obtained when these current models were accepted is recorded and thereafter used as $p(r|\mathbf{g}, \mathbf{V}, \mathbf{a})$ in the calculation of the acceptance ratio. After training has been completed upon the algorithm reaching the final array row, actions are determined during 100 testing episodes by calculating

$$P(a_{\text{new}} = 1|\mathbf{V}, \mathbf{r}) = \frac{1}{\text{Tot}}\sum_{n=\beta}^{N-1} C_n(\mathbf{V}_{\text{new}} \cdot \mathbf{g}_n) \quad (13)$$

for each array at each simulation time step. The summation considers only rows greater than index β, which determines the number of rows discarded to account for the burn-in period. The variable Tot is the sum of all row counter values recorded after the burn-in period. The array with the largest response at each time step determines the action taken during inference in a winner-take-all fashion. It should be noted that, although we have used the notation $p(r|\mathbf{g}, \mathbf{V}, \mathbf{a})$ for consistency with the rest of this Article, this quantity is not a probability, but is in fact a reward.

Neural network benchmark models. To benchmark the performance of RRAM-based MCMC sampling against a state-of-the-art machine learning approach, two neural network models were implemented using the TensorFlow Python library and applied to the same tasks. Both of these models were composed of a total number of synapses equal to the number of differential conductance pairs in the memory arrays (4,096 in both cases). Both models were three-layer feed-forward neural networks using 32-bit floating-point precision synaptic weights. The hidden layer of each model was sized such that the number of synaptic weights in the network was equal to 4,096. The size of the first layer is consistent with the number of input features. In the supervised neural network, the hidden and output units were logistic functions, as this is the function used in our logistic regression model. The model was optimized by minimizing the categorical cross-entropy in the training dataset over 100 training epochs with the adaptive moment estimation (Adam) optimization algorithm. In the reinforcement learning model, the hidden and output units were non-rectifying linear units, because this corresponded to the row function block used in our reinforcement learning model. The parameters of the models were optimized by minimizing the mean-squared loss using the experience replay technique[47], also using the Adam optimization algorithm.

Data availability

The Wisconsin breast cancer dataset[41], the MIT-BIH ECG dataset[42] and the reinforcement learning simulation environment are publicly available. All other measured data are freely available upon request.

Code availability

All software programs used in the presentation of the Article are freely available upon request.

Received: 24 April 2020; Accepted: 30 November 2020;
Published online: 18 January 2021

References

1. Shi, W., Cao, J., Zhang, Q., Li, Y. & Xu, L. Edge computing: vision and challenges. *IEEE Internet Things J.* **3**, 637–646 (2016).
2. *Edge AI Chipsets: Technology Outlook and Use Cases*. Technical Report (ABI Research, 2019).
3. von Neumann, J. First draft of a report on the EDVAC. *IEEE Ann. Hist. Comput.* **15**, 27–75 (1993).
4. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
5. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
6. Strubell, E., Ganesh, A. & Mccallum, A. Energy and policy considerations for deep learning in NLP. In *Proc. 57th Annual Meeting of the Association for Computational Linguistics* (*ACL*) 3645–3650 (ACL, 2019).
7. Li, D., Chen, X., Becchi, M. & Zong, Z. Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs. In *2016 IEEE International Conferences on Big Data and Cloud Computing* (*BDCloud*), *Social Computing and Networking* (*SocialCom*), *Sustainable Computing and Communications* (*SustainCom*) (*BDCloud-SocialCom-SustainCom*) 477–484 (IEEE, 2016).
8. Chua, L. Memristor—the missing circuit element. *IEEE Trans. Circuit Theory* **18**, 507–519 (1971).
9. Prezioso, M. et al. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61–64 (2014).
10. Wong, H. P. et al. Phase change memory. *Proc. IEEE* **98**, 2201–2227 (2010).
11. Chappert, C., Fert, A. & Dau, F. The emergence of spin electronics in data storage. *Nat. Mater.* **6**, 813–823 (2007).
12. Liu, Q. et al. Resistive switching: real-time observation on dynamic growth/dissolution of conductive filaments in oxide-electrolyte-based ReRAM. *Adv. Mater.* **24**, 1774–1774 (2012).
13. Beck, A., Bednorz, J. G., Gerber, C., Rossel, C. & Widmer, D. Reproducible switching effect in thin oxide films for memory applications. *Appl. Phys. Lett.* **77**, 139–141 (2000).
14. Garbin, D. et al. HfO_2-based OxRAM devices as synapses for convolutional neural networks. *IEEE Trans. Electron Dev.* **62**, 2494–2501 (2015).
15. Gokmen, T., Onen, M. & Haensch, W. Training deep convolutional neural networks with resistive cross-point devices. *Front. Neurosci.* **11**, 538 (2017).
16. Boybat, I. et al. Neuromorphic computing with multi-memristive synapses. *Nat. Commun.* **9**, 2514 (2017).
17. Ambrogio, S. et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* **558**, 60–67 (2018).
18. Nandakumar, S. R. et al. Mixed-precision architecture based on computational memory for training deep neural networks. In *Proc. 2018 IEEE International Symposium on Circuits and Systems* (*ISCAS*) 1–5 (IEEE, 2018).
19. Li, C. et al. Efficient and self-adaptive in-situ learning in multilayer memristor neural networks. *Nat. Commun.* **9**, 2385 (2018).
20. Wang, Z. et al. Reinforcement learning with analogue memristor arrays. *Nat. Electron.* **2**, 115–124 (2019).
21. Yao, P. et al. Fully hardware-implemented memristor convolutional neural network. *Nature* **577**, 641–646 (2020).
22. Burr, G. W. et al. Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses) using phase-change memory as the synaptic weight element. *IEEE Trans. Electron Dev.* **62**, 3498–3507 (2015).
23. Sebastian, A., Krebs, D., Gallo, M. L., Pozidis, H. & Eleftheriou, E. A collective relaxation model for resistance drift in phase change memory cells. In *Proc. 2015 IEEE International Reliability Physics Symposium* MY.5.1–MY.5.6 (IEEE, 2015).
24. Ambrogio, S. et al. Statistical fluctuations in HfO_x resistive-switching memory: Part 1—set/reset variability. *IEEE Trans. Electron Dev.* **61**, 2912–2919 (2014).
25. Sidler, S. et al. Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: impact of conductance response. In *Proc. 2016 46th European Solid-State Device Research Conference* (*ESSDERC*) 440–443 (IEEE, 2016).
26. Agarwal, S. et al. Resistive memory device requirements for a neural algorithm accelerator. In *Proc. 2016 International Joint Conference on Neural Networks* (*IJCNN*) 929–938 (IEEE, 2016).
27. Querlioz, D., Bichler, O., Dollfus, P. & Gamrat, C. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Trans. Nanotechnol.* **12**, 288–295 (2013).
28. Serb, A. et al. Unsupervised learning in probabilistic neural networks with multi-state metal–oxide memristive synapses. *Nat. Commun.* **7**, 12611 (2016).

29. Dalgaty, T. et al. Hybrid neuromorphic circuits exploiting non-conventional properties of RRAM for massively parallel local plasticity mechanisms. *APL Mater.* **7**, 081125 (2019).
30. Balatti, S., Ambrogio, S., Wang, Z. & Ielmini, D. True random number generation by variability of resistive switching in oxide-based devices. *IEEE J. Emerg. Select. Top. Circuits Syst.* **5**, 214–221 (2015).
31. Vodenicarevic, D. et al. Low-energy truly random number generation with superparamagnetic tunnel junctions for unconventional computing. *Phys. Rev. Appl.* **8**, 054045 (2017).
32. Faria, R., Camsari, K. Y. & Datta, S. Implementing Bayesian networks with embedded stochastic MRAM. *AIP Adv.* **8**, 045101 (2018).
33. Mizrahi, A. et al. Neural-like computing with populations of superparamagnetic basis functions. *Nat. Commun.* **9**, 1533 (2018).
34. Camsari, K. Y., Faria, R., Sutton, B. M. & Datta, S. Stochastic *p*-bits for invertible logic. *Phys. Rev. X* **7**, 031014 (2017).
35. Borders, W. A. et al. Integer factorization using stochastic magnetic tunnel junctions. *Nature* **573**, 390–393 (2019).
36. Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97–109 (1970).
37. Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature* **521**, 452–459 (2015).
38. Neal, R. M. *Bayesian Learning for Neural Networks* Vol. 118 (Springer Science & Business Media, 2012).
39. Grossi, A. et al. Resistive RAM endurance: array-level characterization and correction techniques targeting deep learning applications. *IEEE Trans. Electron Dev.* **66**, 1281–1288 (2019).
40. Ielmini, D. Modeling the universal set/reset characteristics of bipolar RRAM by field- and temperature-driven filament growth. *IEEE Trans. Electron Dev.* **58**, 4309–4317 (2011).
41. Wolberg, W. H. & Mangasarian, O. L. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proc. Natl Acad. Sci. USA* **87**, 9193–9196 (1990).
42. Moody, G. B. & Mark, R. G. The impact of the MIT-BIH arrhythmia database. *IEEE Eng. Med. Biol. Mag.* **20**, 45–50 (2001).
43. Sutton, R. S. & Barto, A. G. *Introduction to Reinforcement Learning* (MIT Press, 1998).
44. Hoffman, M., Doucet, A., Freitas, N. D. & Jasra, A. Trans-dimensional MCMC for Bayesian policy learning. In *Proc. 20th International Conference on Neural Information Processing Systems*, *NIPS'07* 665–672 (Curran Associates, 2007).
45. Barto, A. G., Sutton, R. S. & Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cybern.* **SMC-13**, 834–846 (1983).
46. Berdan, R. et al. In-memory reinforcement learning with moderately-stochastic conductance switching of ferroelectric tunnel junctions. In *Proc. 2019 Symposium on VLSI Technology* T22–T23 (IEEE, 2019).
47. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
48. Pourret, O., Naïm, P. & Marcot, B. *Bayesian Networks*: *A Practical Guide to Applications* (Wiley, 2008).
49. Maclaurin, D. & Adams, R. P. Firefly Monte Carlo: exact MCMC with subsets of data. In *Proc. Thirtieth Conference on Uncertainty in Artificial Intelligence*, *UAI'14* 543–552 (AUAI Press, 2014).
50. Korattikara, A., Chen, Y. & Welling, M. Austerity in MCMC land: cutting the Metropolis–Hastings budget. In *Proc. 31st International Conference on Machine Learning* 181–189 (PMLR, 2014).
51. Hoffman, M. D. & Gelman, A. The no-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.* **15**, 1593–1623 (2014).
52. Liu, H. & Setiono, R. Chi2: feature selection and discretization of numeric attributes. In *Proc. 7th IEEE International Conference on Tools with Artificial Intelligence* 388–391 (IEEE, 1995).

Acknowledgements

We acknowledge funding support from the French ANR via Carnot funding as well as the H2020 MeM-Scales project (871371) and the European Research Council (grant NANOINFER, no. 715872). In addition, we thank E. Esmanhotto, J. Sandrini and C. Cagli (CEA-Leti) for help with the measurement set-up, J. F. Nodin (CEA-Leti) for providing the images in Fig. 3d and to S. Mitra (Stanford University), M. Payvand (ETH Zurich), A. Valentian, M. Solinas-Angel, E. Nowak (CEA-Leti), J. Diard (CNRS, Université Grenoble Alpes), P. Bessiére and J. Droulez (CNRS, Sorbonne Université), J. Grollier (CNRS, Thales) and J.-M. Portal (Aix-Marseille Université) for discussing various aspects of the Article.

Author contributions

T.D. developed the concept of RRAM-based MCMC sampling. N.C. built the computer-in-the-loop test set-up with the resistive memory array. T.D. and N.C. performed the computer-in-the-loop experiments with the resistive memory array. T.D. implemented the behavioural simulator, performed measurements on the array, which were used to calibrate the simulation and performed the benchmarking. K.-E.H., C.T. and D.Q. performed the design and energy analysis of the full system implementation. T.D., D.Q. and E.V. developed ideas and wrote the Article together.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41928-020-00523-3.

Correspondence and requests for materials should be addressed to T.D., D.Q. or E.V.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Bibliography

[1] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis, 2023.

[2] Karl Rupp. Microprocessor trend data, Feb 2018.

[3] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[4] Abu Sebastian, Manuel Le Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. Memory devices and applications for in-memory computing. *Nature nanotechnology*, 15(7):529–544, 2020.

[5] Daniele Ielmini and H-S Philip Wong. In-memory computing with resistive switching devices. *Nature Electronics*, 1(6):333, 2018.

[6] Seungchul Jung, Hyungwoo Lee, Sungmeen Myung, Hyunsoo Kim, Seung Keun Yoon, Soon-Wan Kwon, Yongmin Ju, Minje Kim, Wooseok Yi, Shinhee Han, et al. A crossbar array of magnetoresistive memory devices for in-memory computing. *Nature*, 601(7892):211–216, 2022.

[7] Prabhat Kumar Gupta and Ramdas Kumaresan. Binary multiplication with pn sequences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(4):603–606, 1988.

[8] Tifenn Hirtzlin, Marc Bocquet, Bogdan Penkovsky, Jacques-Olivier Klein, Etienne Nowak, Elisa Vianello, Jean-Michel Portal, and Damien Querlioz. Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays. *Frontiers in neuroscience*, 13:1383, 2020.

[9] Castets-Renard Céline. *Protection des données personnelles et intelligence artificielle*. Claude Blumann éd., Annuaire de droit de l'Union européenne. 2021. Éditions Panthéon-Assas, 2021.

[10] https://www.upenergie.com/marche-energie-infographie-consommation-par-habitant-et-par-ville-delectricite-en-france/.

[11] K-E Harabi, Clement Turck, Marie Drouhin, Adrien Renaudineau, T Bersani-Veroni, D Querlioz, T Hirtzlin, E Vianello, M Bocquet, and J-M Portal. A multimode hybrid memristor-cmos prototyping platform supporting digital and analog projects. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, pages 184–185, 2023.

[12] Fadi Jebali, Atreya Majumdar, Clément Turck, Kamel-Eddine Harabi, Mathieu-Coumba Faye, Eloi Muhr, Jean-Pierre Walder, Oleksandr Bilousov, Amadeo Michaud, Elisa Vianello, et al. Powering ai at the edge: A robust, memristor-based binarized neural network with near-memory computing and miniaturized solar cell. *arXiv preprint arXiv:2305.12875*, 2023.

[13] Thomas Dalgaty, Niccolo Castellani, Clément Turck, Kamel-Eddine Harabi, Damien Querlioz, and Elisa Vianello. In situ learning using intrinsic memristor variability via markov chain monte carlo sampling. *Nature Electronics*, 4(2):151–161, 2021.

[14] Nicholas Enticknap. Von neumann and the origins of the digital computer. *IEEE Annals of the History of Computing*, 36(4):28–31, 2014.

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

[16] Robert R. Gilruth. To the moon and beyond. *The Aeronautical Journal*, 75(721):1–17, 1971.

[17] John Bardeen and Walter Hauser Brattain. The transistor, a semi-conductor triode. *Physical Review*, 74(2):230, 1948.

[18] Michael Riordan, Lillian Hoddeson, and Conyers Herring. The invention of the transistor. *Reviews of Modern Physics*, 71(2):S336, 1999.

[19] Charles Kittel. Introduction to solid state physics eighth edition. 2021.

[20] William F Brinkman, Douglas E Haggan, and William W Troutman. A history of the invention of the transistor and where it will lead us. *IEEE journal of solid-state circuits*, 32(12):1858–1865, 1997.

[21] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[22] Chris A Mack. Fifty years of moore's law. *IEEE Transactions on semiconductor manufacturing*, 24(2):202–207, 2011.

[23] Scott E Thompson, Robert S Chau, Tahir Ghani, Kaizad Mistry, Sunit Tyagi, and Mark T Bohr. In search of" forever," continued transistor scaling one new material at a time. *IEEE Transactions on semiconductor manufacturing*, 18(1):26–36, 2005.

[24] Simon M Sze, Yiming Li, and Kwok K Ng. *Physics of semiconductor devices*. John wiley & sons, 2021.

[25] Scott E Thompson and Srivatsan Parthasarathy. Moore's law: the future of si microelectronics. *Materials today*, 9(6):20–25, 2006.

[26] JA Hoerni. Patents [method of manufacturing semiconductor devices-us patent no. 3,025,589]. *IEEE Solid-State Circuits Society Newsletter*, 12(2):41–42, 2007.

[27] Sorin Cristoloveanu. Silicon on insulator technologies and devices: from present to future. *Solid-State Electronics*, 45(8):1403–1411, 2001.

[28] R Carter, J Mazurier, L Pirro, JU Sachse, P Baars, J Faul, C Grass, G Grasshoff, P Javorka, T Kammler, et al. 22nm fdsoi technology for emerging mobile, internet-of-things, and rf applications. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 2–2. IEEE, 2016.

[29] Daniel Pham, Larry Larson, and Ji-Woon Yang. Finfet device junction formation challenges. In *2006 International Workshop on Junction Technology*, pages 73–77. IEEE, 2006.

[30] Jean-Pierre Colinge et al. *FinFETs and other multi-gate transistors*, volume 73. Springer, 2008.

[31] TSMC. Tsmc holds 3nm volume production and capacity expansion ceremony, marking a key milestone for advanced manufacturing, 12 2022.

[32] Krutideepa Bhol, Biswajit Jena, and Umakanta Nanda. Journey of mosfet from planar to gate all around: A review. *Recent Patents on Nanotechnology*, 16(4):326–332, 2022.

[33] Samsung. Optimized 3nm process achieves 45% reduced power usage, 23% improved performance and 16% smaller surface area compared to 5nm process, June 2022.

[34] McKinsey & Company. Semiconductor design and manufacturing: Achieving leading-edge capabilities, August 2020.

[35] McKinsey & Company. The chips and science act: Here's what's in it, October 2022.

[36] Richard Waters. Can intel become the chip champion the us needs?, APRIL 2023.

[37] Yanna Luo, Qingzhu Zhang, Lei Cao, Weizhuo Gan, Haoqing Xu, Yu Cao, Jie Gu, Renren Xu, Gangping Yan, Jiali Huo, Zhenhua Wu, and Huaxiang Yin. Investigation of novel hybrid channel complementary fet scaling beyond 3-nm node from device to circuit. *IEEE Transactions on Electron Devices*, 69:3581–3588, 06 2022.

[38] Julien Ryckaert, Pieter Schuddinck, Pieter Weckx, Guillaume Bouche, Benjamin Vincent, J. Smith, Yasser Sherazi, Arindam Mallik, Hans Mertens, Steven Demuynck, Trong Huynh Bao, Anabela Veloso, Naoto Horiguchi, Anda Mocuta, Dan Mocuta, and Juergen Boemmels. The complementary fet (cfet) for cmos scaling beyond n3. *2018 IEEE Symposium on VLSI Technology*, pages 141–142, 2018.

[39] Andrew B Kahng, Jens Lienig, Igor L Markov, and Jin Hu. *VLSI physical design: from graph partitioning to timing closure*. Springer Science & Business Media, 2011.

[40] Pong P Chu. *RTL hardware design using VHDL: coding for efficiency, portability, and scalability*. John Wiley & Sons, 2006.

[41] Xiaoqing Xu, Nishi Shah, Andrew Evans, Saurabh Sinha, Brian Cline, and Greg Yeric. Standard cell library design and optimization methodology for asap7 pdk. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 999–1004. IEEE, 2017.

[42] Jan Rabaey. Restoring the magic in design. In *2023 9th International Workshop on Advances in Sensors and Interfaces (IWASI)*, pages 6–6, 2023.

[43] Scott McCartney. *ENIAC: The triumphs and tragedies of the world's first computer*. Walker & Company, 1999.

[44] J. von Neumann. First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4):27–75, 1993.

[45] John W Backus. The ibm 701 speedcoding system. *Journal of the ACM (JACM)*, 1(1):4–6, 1954.

[46] Gene M Amdahl, Gerrit A Blaauw, and Frederick P Brooks. Architecture of the ibm system/360. *IBM Journal of Research and Development*, 8(2):87–101, 1964.

[47] Robert Noyce and Marcian Hoff. A history of microprocessor development at intel. *IEEE Micro*, 1(01):8–21, 1981.

[48] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.

[49] John D Owens, Mike Houston, David Luebke, Simon Green, John E Stone, and James C Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[50] Chang-Chi Lee, CP Hung, Calvin Cheung, Ping-Feng Yang, Chin-Li Kao, Dao-Long Chen, Meng-Kai Shih, Chien-Lin Chang Chien, Yu-Hsiang Hsiao, Li-Chieh Chen, et al. An overview of the development of a gpu with integrated hbm on silicon interposer. In *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pages 1439–1444. IEEE, 2016.

[51] Jeff Dean, David Patterson, and Cliff Young. A new golden age in computer architecture: Empowering the machine-learning revolution. *IEEE Micro*, 38(2):21–29, 2018.

[52] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proc. ISCA*, pages 1–12. IEEE, 2017.

[53] John Backus. Can programming be liberated from the von neumann style? a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641, 1978.

[54] Ardavan Pedram, Stephen Richardson, Mark Horowitz, Sameh Galal, and Shahar Kvatinsky. Dark memory and accelerator-rich system optimization in the dark silicon era. *IEEE Design & Test*, 34(2):39–50, 2017.

[55] M Mitchell Waldrop. The chips are down for moore's law. *Nature News*, 530(7589):144, 2016.

[56] D. Querlioz, O. Bichler, A.F. Vincent, and C. Gamrat. Bioinspired Programming of Memory Devices for Implementing an Inference Engine. *Proc. IEEE*, 103(8):1398–1416, 2015.

[57] JB Furness. Types of neurons in the enteric nervous system. *Journal of the autonomic nervous system*, 81(1-3):87–96, 2000.

[58] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[59] Hugh R Wilson and Jack D Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12(1):1–24, 1972.

[60] William H Calvin and CHARLES F Stevens. Synaptic noise and other sources of randomness in motoneuron interspike intervals. *Journal of neurophysiology*, 31(4):574–587, 1968.

[61] Semir Zeki. A massively asynchronous, parallel brain. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 370(1668):20140174, 2015.

[62] Simon J Thorpe. Spike arrival times: A highly efficient coding scheme for neural networks. In *Parallel processing in neural systems*, pages 91–94, 1990.

[63] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *bioRxiv*, 2020.

[64] Tifenn Hirtzlin. *Digital Implementation of Neuromorphic systems using Emerging Memory devices*. Theses, Université Paris-Saclay, November 2020.

[65] Giacomo Indiveri and Shih-Chii Liu. Memory and information processing in neuromorphic systems. *Proc. IEEE*, 103(8):1379–1397, 2015.

[66] Russell Reed and Robert J Marks. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. The MIT Press, 02 1999.

[67] Unsupervised Learning. In *Unsupervised Learning: Foundations of Neural Computation*. The MIT Press, 05 1999.

[68] Gagandeep Singh, Lorenzo Chelini, Stefano Corda, Ahsan Javed Awan, Sander Stuijk, Roel Jordans, Henk Corporaal, and Albert-Jan Boonstra. Near-memory computing: Past, present, and future. *Microprocessors and Microsystems*, 71:102868, 2019.

[69] Naveen Verma, Hongyang Jia, Hossein Valavi, Yinqi Tang, Murat Ozatay, Lung-Yen Chen, Bonan Zhang, and Peter Deaville. In-memory computing: Advances and prospects. *IEEE Solid-State Circuits Magazine*, 11(3):43–55, 2019.

[70] Kwanyeob Chae, Jiyeon Park, Jaegeun Song, Billy Koo, Jihun Oh, Shinyoung Yi, Won Lee, Dongha Kim, Taekyung Yeo, Kyeongkeun Kang, Sangsoo Park, Eunsu Kim, Sukhyun Jung, Sanghune Park, Sungcheol Park, Mijung Noh, Hyogyuem Rhew, and Jongshin Shin. A 4nm 1.15tb/s hbm3 interface with resistor-tuned offset-calibration and in-situ margin-detection. In *2023 IEEE International Solid- State Circuits Conference (ISSCC)*, pages 1–3, 2023.

[71] Leon Chua. Memristor-the missing circuit element. *IEEE Transactions on circuit theory*, 18(5):507–519, 1971.

[72] Dmitri Strukov, Gregory Snider, Duncan Stewart, and Stan Williams. The missing memristor found. *Nature*, 453:80–3, 06 2008.

[73] Sabina Spiga, Abu Sebastian, Damien Querlioz, and Bipin Rajendran. *Memristive Devices for Brain-Inspired Computing: From Materials, Devices, and Circuits to Applications-Computational Memory, Deep Learning, and Spiking Neural Networks*. Woodhead Publishing, 2020.

[74] Bernard Dieny, Ioan Lucian Prejbeanu, Kevin Garello, Pietro Gambardella, Paulo Freitas, Ronald Lehndorff, Wolfgang Raberg, Ursula Ebels, Sergej O Demokritov, Johan Akerman, et al. Opportunities and challenges for spintronics in the microelectronics industry. *Nature Electronics*, 3(8):446–459, 2020.

[75] H-S Philip Wong, Heng-Yuan Lee, Shimeng Yu, Yu-Sheng Chen, Yi Wu, Pang-Shiu Chen, Byoungil Lee, Frederick T Chen, and Ming-Jinn Tsai. Metal–oxide rram. *Proceedings of the IEEE*, 100(6):1951–1970, 2012.

[76] H-S Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P Reifenberg, Bipin Rajendran, Mehdi Asheghi, and Kenneth E Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, 2010.

[77] Thomas Mikolajick, Stefan Slesazeck, Min Hyuk Park, and Uwe Schroeder. Ferroelectric hafnium oxide for ferroelectric random-access memories and ferroelectric field-effect transistors. *Mrs Bulletin*, 43(5):340–346, 2018.

[78] Shimeng Yu. Neuro-inspired computing with emerging nonvolatile memories. *Proc. IEEE*, 106(2):260–285, 2018.

[79] Lingyun Shi, Guohao Zheng, Bobo Tian, Brahim Dkhil, and Chungang Duan. Research progress on solutions to the sneak path issue in memristor crossbar arrays. *Nanoscale Adv.*, 2:1811–1827, 2020.

[80] Stefano Ambrogio, Pritish Narayanan, Hsinyu Tsai, Robert M Shelby, Irem Boybat, Carmelo Nolfo, Severin Sidler, Massimo Giordano, Martina Bodini, Nathan CP Farinha, et al. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, 558(7708):60, 2018.

[81] Mirko Prezioso, Farnood Merrikh-Bayat, Brian Hoskins, Gina Adam, Konstantin Likharev, and Dmitri Strukov. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, 521, 12 2014.

[82] Zhongrui Wang, Saumil Joshi, Sergey Savel'ev, Wenhao Song, Rivu Midya, Yunning Li, Mingyi Rao, Peng Yan, Shiva Asapu, Ye Zhuo, et al. Fully memristive neural networks for pattern classification with unsupervised learning. *Nature Electronics*, 1(2):137, 2018.

[83] Cheng-Xin Xue, Yen-Cheng Chiu, Ta-Wei Liu, Tsung-Yuan Huang, Je-Syu Liu, Ting-Wei Chang, Hui-Yao Kao, Jing-Hong Wang, Shih-Ying Wei, Chun-Ying Lee, et al. A cmos-integrated compute-in-memory macro based on resistive random-access memory for ai edge devices. *Nature Electronics*, 4(1):81–90, 2021.

[84] Samuel D. Spetalnick, Muya Chang, Shota Konno, Brian Crafton, Ashwin S. Lele, Win-San Khwa, Yu-Der Chih, Meng-Fan Chang, and Arijit Raychowdhury. A 2.38 mcells/mm2 9.81 -350 tops/w rram compute-in-memory macro in 40nm cmos with hybrid offset/ioff cancellation and icell rblsl drop mitigation. In *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, pages 1–2, 2023.

[85] Kamel-Eddine Harabi, Tifenn Hirtzlin, Clément Turck, Elisa Vianello, Raphaël Laurent, Jacques Droulez, Pierre Bessière, Jean-Michel Portal, Marc Bocquet, and Damien Querlioz. A memristor-based bayesian machine. *Nature Electronics*, 6(1):52–63, 2023.

[86] Elise Payzan-LeNestour, Simon Dunne, Peter Bossaerts, and John P. O'Doherty. The neural representation of unexpected uncertainty during value-based decision making. *Neuron*, 79(1):191–201, 2013.

[87] Elise Payzan-LeNestour and Peter Bossaerts. Do not bet on the unknown versus try to find out more: Estimation uncertainty and "unexpected uncertainty" both modulate exploration. *Frontiers in Neuroscience*, 6, 2012.

[88] Brian R Gaines. Stochastic computing systems. In *Advances in information systems science*, pages 37–172. Springer, 1969.

[89] Armin Alaghi and John P Hayes. Survey of stochastic computing. *ACM TECS*, 12(2s):92, 2013.

[90] Chris Winstead. Tutorial on stochastic computing. In *Stochastic Computing: Techniques and Applications*, pages 39–76. Springer, 2019.

[91] Arun Rai. Explainable ai: From black box to glass box. *Journal of the Academy of Marketing Science*, 48(1):137–141, 2020.

[92] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[93] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[94] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[95] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[96] Weisheng Zhao, Claude Chappert, Virgile Javerliac, and Jean-Pierre Noziere. High speed, high stability and low power sensing amplifier for mtj/cmos hybrid logic circuits. *IEEE Transactions on Magnetics*, 45(10):3784–3787, 2009.

[97] Weisheng Zhao, Mathieu Moreau, Erya Deng, Yue Zhang, Jean-Michel Portal, Jacques-Olivier Klein, Marc Bocquet, Hassen Aziza, Damien Deleruyelle, Christophe Muller, et al. Synchronous non-volatile logic gate design based on resistive switching memories. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(2):443–454, 2014.

[98] Pierre Bessière, Emmanuel Mazer, Juan Manuel Ahuactzin, and Kamel Mekhnacha. *Bayesian programming*. CRC press, 2013.

[99] Alessandro Grossi, Cristian Zambelli, Piero Olivo, Alberto Crespo-Yepes, Javier Martin-Martinez, Rosana Rodríguez, Monserrat Nafria, Eduardo Perez, and Christian Wenger. Electrical characterization and modeling of 1t-1r rram arrays with amorphous and polycrystalline hfo2. *Solid-State Electronics*, 128:187–193, 2017.

[100] Meng-Fan Chang, Che-Wei Wu, Chia-Cheng Kuo, Shin-Jang Shen, Ku-Feng Lin, Shu-Meng Yang, Ya-Chin King, Chorng-Jung Lin, and Yu-Der Chih. A 0.5 v 4mb logic-process compatible embedded resistive ram (reram) in 65nm cmos using low-voltage current-mode sensing scheme with 45ns random read time. In *2012 IEEE International Solid-State Circuits Conference*, pages 434–436. IEEE, 2012.

[101] Pete Warden and Daniel Situnayake. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, 2019.

[102] Damir Vodenicarevic, Nicolas Locatelli, Alice Mizrahi, Joseph S Friedman, Adrien F Vincent, Miguel Romera, Akio Fukushima, Kay Yakushiji, Hitoshi Kubota, Shinji Yuasa, et al. Low-energy truly random number generation with superparamagnetic tunnel junctions for unconventional computing. *Phys. Rev. Appl.*, 8(5):054045, 2017.

[103] Rafatul Faria, Kerem Y Camsari, and Supriyo Datta. Implementing bayesian networks with embedded stochastic mram. *AIP Advances*, 8(4):045101, 2018.

[104] Kerem Y Camsari, Brian M Sutton, and Supriyo Datta. P-bits for probabilistic spin logic. *Applied Physics Reviews*, 6(1):011305, 2019.

[105] William A Borders, Ahmed Z Pervaiz, Shunsuke Fukami, Kerem Y Camsari, Hideo Ohno, and Supriyo Datta. Integer factorization using stochastic magnetic tunnel junctions. *Nature*, 573(7774):390–393, 2019.

[106] Rekha Govindaraj, Swaroop Ghosh, and Srinivas Katkoori. Csro-based reconfigurable true random number generator using rram. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(12):2661–2670, 2018.

[107] Akio Fukushima, Takayuki Seki, Kay Yakushiji, Hitoshi Kubota, Hiroshi Imamura, Shinji Yuasa, and Koji Ando. Spin dice: A scalable truly random number generator based on spintronics. *Applied Physics Express*, 7(8):083001, 2014.

[108] Simone Balatti, Stefano Ambrogio, Zhongqiang Wang, and Daniele Ielmini. True random number generation by variability of resistive switching in oxide-based devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 5(2):214–221, 2015.

[109] Leonel Sousa. Nonconventional computer arithmetic circuits, systems and applications. *IEEE Circ. Syst. Magazine*, 21(1):6–40, 2021.

[110] Stephanie Devuyst, T Dutoit, and M Kerkhofs. The dreams databases and assessment algorithm. [data set]. zenodo. https://doi.org/10.5281/zenodo.2650142. *Zenodo: Geneva, Switzerland*, 2005.

[111] Antonio Pullini, Davide Rossi, Igor Loi, Alfio Di Mauro, and Luca Benini. Mr. wolf: A 1 gflop/s energy-proportional parallel ultra low power soc for iot edge processing. In *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*, pages 274–277, 2018.

[112] Kamel-Eddine HARABI. *Energy Efficient Memristor-Based Artificial Intelligence Accelerators using In/Near Memory Computing*. Theses, Université Paris-Saclay, July 2023.

[113] A. Majumdar et al. Model of the weak reset process in hfo x resistive memory for deep learning frameworks. *IEEE Trans. Elect. Dev.*, 68:4925, 2021.