UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

Réseaux neuronaux robustes face à la variabilité de l'apprentissage machine sur crossbar passif de mémoires résistives à base de TiO_2

Mémoire de maîtrise
Spécialité : génie électrique

Philippe Gabriel DROLET

Sherbrooke (Québec) Canada

Avril 2023

MEMBRES DU JURY

Dominique DROUIN

Directeur

Yann BEILLIARD

Codirecteur

Sean WOOD

Évaluateur

Fabien ALIBART

Évaluateur

RÉSUMÉ

La présence des réseaux neuronaux dans notre quotidien connaît une augmentation exponentielle. Les réseaux sociaux, les moteurs de recherche et le commerce électronique ne sont que quelques exemples de domaines les sollicitant en permanence. La taille de ces réseaux, à l'image de leur présence, a également connu un constant essor.

Or, plus un réseau neuronal comporte de paramètres, plus il consomme d'énergie puisqu'il nécessite toujours plus d'accès successifs en mémoire afin de chercher et/ou d'altérer ceux-ci. C'est ce goulot d'étranglement entre le processeur et les données qui limite dramatiquement l'efficience énergétique des réseaux neuronaux implémentés sur l'architecture de von Neumann. Une approche prometteuse pour détourner ce problème est l'utilisation d'une composante électronique permettant une version analogue des opérations de multiplication matrice-vecteur essentielles à l'apprentissage machine directement sur mémoire : les mémoires résistives. Cependant, les défauts inhérents dont celles-ci sont affligées les rendent difficiles à utiliser en tant que poids synaptique discret et précis. Il devient donc très intéressant de considérer certaines sources de variabilité de ces mémoires émergentes lors de l'entraînement des réseaux neuronaux afin d'exploiter leurs facultés d'apprentissage pour les rendre ainsi aptes à classifier efficacement en les utilisant malgré leur nombreuses non-idéalités. Ces réseaux stochastiques sont donc robustes à la variabilité matérielle des dispositifs.

Une enquête du potentiel de réseaux neuronaux considérant ces défauts stochastiques fut menée durant ce projet de maîtrise. De nouvelles techniques de caractérisation de la variabilité des crossbars et des mémoires résistives furent élaborées. L'utilisation de ces données pour injecter du bruit sur les poids synaptiques du réseau lors de son entraînement permet de créer un réseau plus robuste. Il est prouvé en simulation qu'un réseau conscient des non-idéalités est considérablement plus précis qu'un réseau entraîné naïvement pour une même tâche de classification de demi-lunes lorsque les variabilités sont prises en compte.

Mots-clés : ReRAM, Memristors, Apprentissage machine, IA, efficience énergétique, nanoélectronique, Réseaux neuronaux analogues

À ma famille: Sandra, François et Marie-Pier

REMERCIEMENTS

Je souhaite d'abord remercier chaleureusement mon directeur de maîtrise, Dominique Drouin ainsi que mes superviseurs et encadrants Yann Beilliard, Fabien Alibart, Serge Ecoffey et Sean Wood. Ce projet n'aurait pu aboutir sans leur implication exhaustive et sans les réponses rapides et pertinentes qu'ils ont toujours rendus à mes questionnements. L'effort monumentale nécessaire pour diriger et conseiller autant d'étudiants naviguant tous des projets différents mérite d'être salué. Je tiens particulièrement à souligner la liberté que l'on m'a offerte vis-à-vis les décisions concernant mon projet et son déroulement ainsi que le support clé que mon superviseur principal, Yann Beilliard, m'a apporté. Ma curiosité et mes suggestions ont toujours été respectés et j'ai été en mesure d'orienter mon projet dans une direction à la fois alignée avec mes objectifs d'apprentissage personnels (incluant un formidable séjour de recherche à Lille sous la tutelle de Fabien Alibart) et avec ceux du groupe et de ses maintenant nombreux membres.

C'est ensuite à eux que je tiens à adresser ma reconnaissance. Raphaël Dawant, sans qui mes caractérisations seraient demeurées simulations, son assiduité à la fabrication de dispositifs rendît possible les projets de tant de membres du groupe. Pierre-Antoine Mouny, architecte de la fondation des caractérisations de nos mémoires, les études statistiques de mon projet sont une extension de son travail méthodique. Patrick Dufour, expert de l'électronique qui m'a tutellé et intégré dans son impressionnant projet de conception de PCB. Matthieu Valdenaire, apportant un support versatile et clé à l'équipe autant en caractérisation électrique qu'en fabrication. Victor Yon, avec qui j'ai fréquemment discuté de l'aspect réseau neuronal de mon projet et dont la prose scientifique est une référence d'excellence en la matière. Dr. Abdelouadoud El-Mesoudy, avec qui je n'ai malheureusement pas eut la chance de travailler directement mais qui a légué au groupe la base de nos procédés de fabrication. Tout les membres du groupe m'ont aidés et supportés à leur manière dans l'accomplissement de ce projet. Il fut toujours pertinent de discuter de mes avancements avec eux.

Je tiens également à souligner le travail hors pair de tout les employé(es) de soutien et professionnel(les) de recherche qui, tous ensemble, font la colonne vertébrale des projets de recherche du 3IT. Jonathan Vermette et son implacable dévouement au LCSM et aux projets des étudiants le populant. Vincent-Philippe Rhéaume, son impressionnante passion pour l'électronique se traduit par une expertise qu'il rend généreusement accessible aux étudiants confus du 3IT (comme je le fut si souvent). Marie-Josée Gour qui, lors de mes travaux en salle blanche, fut réellement indispensable et omniprésente.

Finalement, je veux remercier ma famille et mes ami(e)s. En si peu de mots ; vous faites de mon existence une expérience valant la peine d'être vécue.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

Acronyme	Définition
0T1R	0 Transistor 1 Resistor
1S1R	1 Selector 1 Resistor
1T1R	1 Transistor 1 Resistor
3IT	Institut Interdisciplinaire d'Innovation Technologique
BbyB	Bayes by Backprop
BL	Bit Lines
BNN	Bayesian Neural Network
CBRAM	Conductive Bridge Random Access Memory
CMOS	Complementary Metal Oxide Semi-conductor
CNN	Convolutional Neural Network
CNRS	Centre National de la Recherche Scientifique
DNN	Deep Neural Network
ELBO	Expected (ou Evidence) Lower Bound Error
FRQNT	Fonds de Recherche du Québec Nature et Technologie
GAN	Generative Adversarial Networks
HANN	Hardware-Aware Neural Network
HRS	High Resistive State
IA	Intelligence artificielle
IM2NP	Institut Matériaux Microélectronique Nanosciences de Provence
INPAQT	Integrated Nanoelectronics and Packaging for AI and Quantum Technologies
KCL	Kirchoff Current Law
KL	Kullback-Leibler
LRS	Low Resistive State
LSTM	Long Short-Term Memory
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
MFVB	Mean-Field Variational Bayes
MLE	Maximum Likelihood Estimation
MLP	Multilayer perceptron
NN	Neural Network
NSERC	Natural Sciences and Engineering Research Council of Canada
PCB	Printed Circuit Board
PECVD	Plasma Enhanced Chemical Vapor Deposition
ReRAM	Resistive Random Access Memory
RNN	Recursive Neural Network
RTN	Random Telegraph Noise
SEM	Scanning Electron Microscopy
SLP	Single Layer Perceptron
VCM	Valence Change Memory
VMM	VVector Marix Multiplication
WL	Word Lines

CHAPITRE 1
INTRODUCTION

> Il semblerait que nous ayons atteint les limites de ce qui est possible d'accomplir avec les ordinateurs. Cependant, il faut faire attention avec de telles déclarations, elles sonneront sûrement idiotes d'ici cinq ans. [1]
>
> *John von Neumann, 1949 (traduit)*

1.1 Contexte

La complexité sans cesse croissante des problèmes d'ingénierie se traduit par une augmentation exponentielle de la quantité de données à traiter. Rapidement, l'IA basée sur des réseaux de neurones artificiels utilisée pour analyser et interpréter celles-ci devient trop inefficace d'un point de vue énergique et de rapidité [2]. La structure du matériel informatique basé sur l'architecture classique von Neumann (séparant mémoire et processeur) a de sévères difficultés à gérer de trop grandes quantités de données. À cause de cette architecture, l'efficacité des opérations est limitée par les nombreux et lents transferts entre la mémoire et le processeur (dans le cas d'un réseau neuronal, pour aller chercher et modifier les paramètres en mémoire), créant ainsi un goulot d'étranglement entre eux. Le problème est tel que l'utilisation du calcul nuagique est devenue une nécessité pour les réseaux lorsqu'ils sont de taille considérable. C'est entre autres pourquoi la quantité de calculs exécutés dans des centres de données aux États-Unis a augmenté de 550% entre 2010 et 2018 [3]. Malgré les nombreuses optimisations dans la répartition et la maximisation de l'utilisation de leurs ressources, ces centres demeurent extrêmement énergivores et partiellement ou complètement incompatibles avec plusieurs systèmes embarqués (e.g : véhicules autonomes et drones) . Il devient donc primordial de trouver de nouvelles méthodes de calcul qui permettront de développer le calcul périphérique basé sur les réseaux de neurones à faible consommation.

Une solution proposée à ce problème est l'utilisation de mémoires résistives (autrement connues sous le nom de memristor). Ces composantes électroniques conceptualisées en 1971 par le Dr L. Chua [4] et puis démontrées physiquement pour la première fois en 2008

par HP Labs [5] ont la capacité à varier leur état de conductance lorsqu'une tension est appliquée sur leurs terminaux. Cette capacité de pair avec leur utilisation en configuration crossbar leur permet de représenter des matrices de poids synaptiques. Une utilisation astucieuse de la loi des noeuds de Kirchoff et de la loi de Ohm permet alors de pratiquer de l'inférence de manière analogue. L'architecture de crossbar de type passif tel que fabriqué à l'Université de Sherbrooke permet une excellente intégration à grande échelle et de récents articles [6] montrent l'efficacité du calcul matriciel sur crossbar passif pour des réseaux neuronaux de taille considérable. Puisque les mémoires résistives sont non-volatiles, que leur consommation énergique et temps de commutation (changement de conductance entre les états de haute résistance (HRS) et de basse résistance (LRS)) sont potentiellement très bas (jusqu'à 10fJ[7] et jusqu'à 85ps par commutation [8]) et qu'elles sont compatibles avec les procédés CMOS, leurs avancées sont surveillées de près dans l'industrie [9] et des produits les utilisant y ont vu le jour[10].

1.2 Déclaration du projet de recherche

Malheureusement, les mémoires résistives présentent encore une grande variabilité rendant leur utilisation difficile dans l'immédiat. Le projet de recherche en question propose une nouvelle approche d'entraînement de réseaux neuronaux prenant ces imperfections en compte. Le type de réseau neuronal sous investigation s'inspire des réseaux neuronaux bayésiens et des réseaux récurrents RNN pour mitiger les pertes de précisions associées à l'apprentissage machine réalisée sur crossbar passif. Le succès de ce projet dépend donc d'une caractérisation et d'une compréhension exhaustive des différentes sources de variabilité qui affligent les mémoires. L'avancée de cette approche et d'autres s'apparentant à celle-ci ouvrira à terme la voie à une panoplie d'applications nécessitant de l'IA à faible consommation énergétique et à grande rapidité. Les mémoires émergentes sont l'une des, sinon la principale avenue pour accomplir ceci. Ce projet chercha donc à associer les malencontreuses propriétés stochastiques des dispositifs tels qu'ils existent présentement et les techniques pré-existantes d'entraînement de réseaux neuronaux tel que l'astuce de reparamétrisation afin de rendre l'inférence sur mémoires résistives plus robustes. Son succès aide à défaire la dépendance de l'apprentissage sur crossbar à des dispositifs parfaitement fonctionnels et facilitera une intégration réaliste de mémoires résistives en architecture crossbar passif à moyen terme.

1.3 Question de recherche

Ce projet de recherche vise à utiliser des outils d'apprentissage machine afin d'améliorer les perfomances d'un réseau neuronal transféré sur crossbar de mémoires résistives. La question de recherche de mon projet peut donc se formuler ainsi :

"Comment pouvons-nous rendre les réseaux neuronaux plus robustes à leur transfert sur crossbar de mémoires résistives à base de TiO_2" ?

1.4 Objectifs du projet de recherche

L'objectif principal du projet fut d'obtenir un type de réseau neuronal robuste se prêtant particulièrement bien au transfert sur mémoires résistives. Le modèle fut conçu avec son efficacité et sa simplicité en tête afin qu'il demeure réalistement utilisable. Cet objectif fut accomplie en prouvant par simulation que le réseau stochastique présente une plus grande précision après plus de transferts simulés qu'un réseau régulier. L'objectif principal peut être divisé en sous-objectifs énumérés ci-dessous. Une liste de méthodologie associé à chaque sous-objectif leur est imbriquée.

1. Déterminer et quantifier la variabilité des mémoires résistives
   - par la caractérisation exhaustive de la variabilité de programmation, de l'échec et des effets stochastiques des mémoires résistives
   - par la conception d'un montage (électrique et logiciel) pour pratiquer de l'inférence et des mesures sur crossbar
2. Intégrer la variabilité des mémoires à un réseau neuronal
   - par l'agencement des techniques classiques d'apprentissage de réseaux neuronaux telles que le *dropout*, la reparamétrisation des poids et la régularisation avec les sources de variabilité des mémoires expérimentalement mesurées
   - par l'adaptation logicielle de la librairie PyTorch [11] pour intégrer les imperfections des mémoires dans l'entraînement du réseau
3. Quantifier l'amélioration des performances en fonction de cette inclusion de la variabilité
   - en entraînant un réseau régulier et en comparant ses performances lorsque la variabilité y est ajoutée à celles d'un réseau stochastique

Quelques avenues de recherche connexes plutôt dissociées de l'objectif principal entreprises lors de cette maîtrise sont également présentées en annexe.

1.5 Contributions originales

La conception d'un nouveau type de réseaux neuronal spécialisée pour son utilisation sur notre technologie de mémoires résistives fut réalisée durant ce projet. Le réseau est plus robuste face à la variabilité qu'un réseau régulier partageant la même structure (il est capable de bien classifier un jeu de données considérablement plus souvent en simulation). Le modèle prend en compte de véritables données expérimentales et réussit à converger vers une solution malgré la variabilité des mémoires. Au niveau de la communauté scientifique, c'est le premier réseau neuronal conscient de plusieurs sources de variabilité différentes à utiliser l'architecture 0T1R, le schéma de polarisation V/3 et ne nécessitant qu'une seule étape de programmation (voir section 2.3) sur des crossbars tuilés. Les méthodes d'ajout de variabilité aux poids synaptiques sont également nouvelles.

Au niveau du groupe de recherche, les principales contributions originales du projet sont : la conception d'un banc de mesures pour crossbar, la création d'une librairie d'entraînement de réseaux neuronaux conscients de la variabilité et polyvalente et la mise en place d'une base de données considérable rapportant le comportement expérimental des mémoires.

1.6 Plan du document

Ce document présente tout d'abord une revue de la littérature concernant les avancées récentes des réseaux neuronaux classiques et stochastiques. la revue de la littérature enchaîne sur les mémoires résistives d'oxyde de titane ainsi qu'une explication basique de leur fonctionnement nécessaire afin de mieux comprendre la réalisation du projet. Elle se termine ensuite par une énumération et comparaison des projets réalisés qui ont servi d'inspiration pour ce projet de maîtrise. Il compare les réseaux neuronaux stochastiques existants et situe le projet de maîtrise par rapport à ceux-ci.

Cette dernière section est suivie d'un article écrit dans le cadre de cette maîtrise présentant les principaux résultats ainsi que la méthodologie de caractérisation et de conception. Finalement, une section perspective présente les prochaines étapes pour l'avancement de cette technique, ses potentielles améliorations et applications. Le tout est suivi d'une conclusion récapitulative et de quelques annexes présentant les projets connexes entrepris en parallèle durant ce projet de maîtrise.

CHAPITRE 2

REVUE DE LA LITTÉRATURE

2.1 Réseaux neuronaux

2.1.1 Fonctionnement et principe

La pierre angulaire des réseaux neuronaux est le neurone formel : une imitation mathématique et fonctionnelle d'un neurone biologique. Celui-ci prend plusieurs valeurs d'entrée (x), les multiplie à un poids synaptique sous forme de matrice (W) (calculée lors de l'apprentissage) en y ajoutant une valeur de biais (B) qui est également une matrice. Le tout est transformé par une fonction d'activation non linéaire (φ), habituellement sigmoïde, pour rendre une donnée de sortie d'un post-neurone (y). Les paramètres du réseau sont ainsi les matrices W et B. L'équation 2.1 illustre cette relation mathématique pour un neurone dont les entrées ne comportent qu'une seule dimension.

$$y = \varphi\left(\sum_{j=0}^{m} W_j x_j + B\right) \tag{2.1}$$

Typiquement, les données d'entrées seront traitées en groupes (*Minibatches*), les paramètres de l'équation deviendront donc y_{ij}, x_{ij}, W_{ij} et B_{ij} et la sommation augmentera d'une dimension. Le perceptron est un algorithme d'apprentissage supervisé qui permet de déterminer mathématiquement les valeurs de paramètres adéquates aux neurones formels et qui garantit une séparation entre deux classes de données (tant qu'elles sont linéairement séparables). Un réseau utilisant les perceptrons en configuration multicouche (MLP) peut exprimer mathématiquement une relation entre ses entrées et ses sorties. La figure 2.1 a est un exemple de perceptron multicouche (à une seule couche cachée). Les limitations du MLP proviennent des difficultés matérielles survenant lorsque la quantité de neurones devient faramineuse.

L'interprétation probabiliste du réseau neuronal se dénote : $P(\mathbf{Y} \mid \mathbf{X}, \mathbf{W}, \mathbf{B})$. Pour un problème de classification, selon une donnée d'entrée $\mathbf{x}$, le réseau attribue une probabilité aux classes de $\mathbf{Y}$ à l'aide des poids synaptiques $\mathbf{W}$ (et des valeurs de biais, $\mathbf{B}$). L'apprentissage des poids synaptiques se fait généralement par MAP (*Maximum a posteriori*) en tentant de maximiser la probabilité que les données d'entrées correspondent à leur étiquette (*label)*

respective selon les poids en question :

$$\begin{aligned} \mathbf{W}^{\text{MAP}} &= \arg\max_{\mathbf{W}} \log P(\mathcal{D} \mid \mathbf{W}) + \log(P(\mathbf{W})) \\ &= \arg\max_{\mathbf{W}} \sum_i \log P\left(\mathbf{y}_i \mid \mathbf{x}_i, \mathbf{W}\right) + \log(P(\mathbf{W})) \end{aligned} \tag{2.2}$$

La descente de gradient assume que l'expression $\log P(\mathcal{D} \mid \mathbf{W})$ est différenciable en w. Par la rétropropagation des gradients, pour chaque poids synaptique, le résultat de cette dérivation (le gradient) est multiplié par une constante α (le taux d'apprentissage) et ensuite additionné à sa valeur précédente. Ceci est répété au travers de multiples itérations (*epochs*) sur le jeu de données. Le logarithme de la distribution à priori des poids $\log(P(\mathbf{w}))$ sert de régularisation. Sans elle, la maximisation des poids ne serait pas MAP mais plutôt MLE (*Maximum Likelihood Approximation*). La régularisation permet de décourager l'apprentissage d'un modèle trop complexe (où il y aurait probablement du surapprentissage). Définir une bonne distribution à priori sur les poids permet également de réduire considérablement la variance d'un modèle sans trop augmenter son biais. [12]

Il y existe plusieurs types de réseaux neuronaux, tels que les réseaux neuronaux à convolution, habituellement utilisés dans le traitement d'images [13, 14], les machines de Boltzmann restreintes utilisés pour inférer les distributions de probabilité des données entrantes [15] et les réseaux antagonistes génératifs (GAN) qui utilisent deux réseaux dont l'entraînement s'accomplit en fonction d'un coût résultant de leur capacité à "l'emporter" l'un sur l'autre dans une compétition [16].

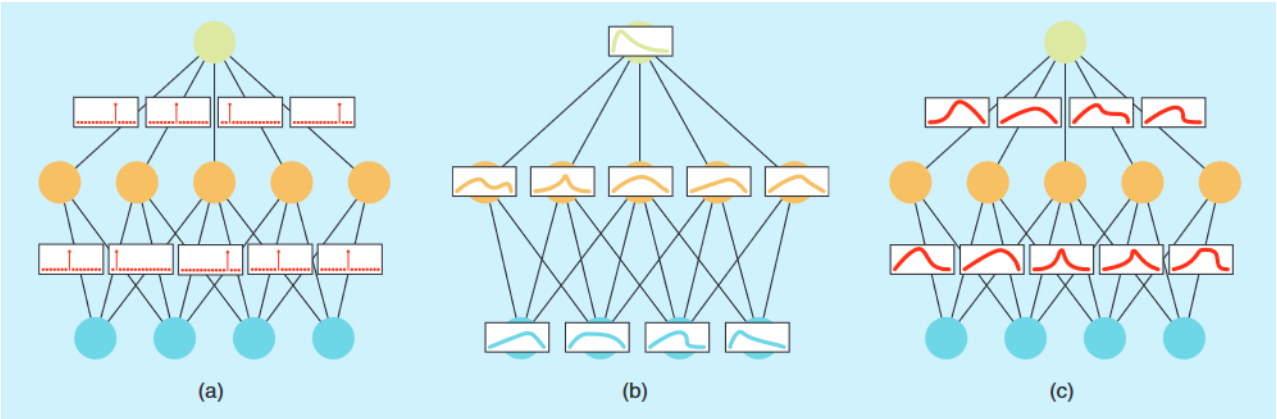

FIGURE 2.1 Reproduit de [17] © [2022] IEEE . a) Illustration d'un MLP classique b) Illustration d'un MLP bayésien où les fonctions d'activations sont représentées par des distributions probabilistes c) Illustration d'un MLP bayésien où les poids synaptiques sont représentés par des distributions probabilistes

2.2 Mémoires résistives

2.2.1 Fonctionnement et principe

Les mémoires résistives (ReRAM) sont des composants électriques passifs à deux terminaux dont la résistance varie en fonction de leur état interne, lui-même modifiable à l'application d'une tension sur leurs électrodes. La raison derrière cette variation de résistance varie en fonction des types de mémoires résistives. Le changement d'état des mémoires filamentaires est dû à la formation et à la destruction d'un filament conducteur à l'intérieur de la couche active de la mémoire résistive. Ce filament est formé de lacunes d'oxygènes conductrices pour les mémoires à changement de valence (VCM) et est formé d'atomes d'une espèce métallique pour les mémoires *conductive bridge* (CBRAM). Dans le cas des mémoires VCM, la couche active est composée d'un oxyde isolant généralement sélectionné parmi les suivants : oxyde d'hafnium (HfOx) [18, 19], alumine (AlOx) [20], oxyde de Nickel (NiOx) [21], oxyde de titane (TiOx) [22] et oxyde de tantale (TaOx) [23]. Les propriétés et les dimensions de cette couche ainsi que celles des électrodes l'emprisonnant influencent les valeurs de résistances élevées et faibles de la mémoire (HRS, LRS), sa rétention, son endurance et sa capacité à être adéquatement mis à l'échelle [24]. Elle est souvent accompagnée d'une mince couche (Al_2O_3 dans [25]) servant à limiter les courants de fuite. L'exploration de la fabrication des mémoires résistives est le sujet de recherche de chercheurs spécialisés à travers le monde.

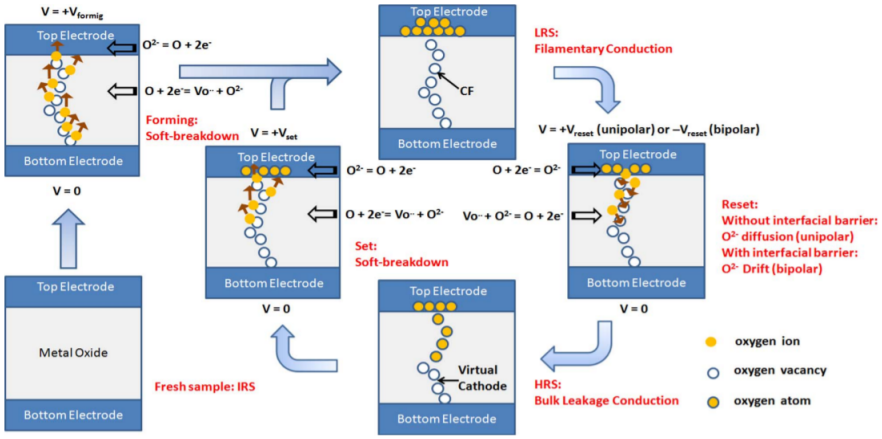

Figure 2.2 Reproduit de [24] © [2012] IEEE.Illustration de la commutation d'une mémoire résistive. Au LRS, le filament conducteur est à sa taille maximale jusqu'à ce qu'une tension négative (dans le cas de mémoires bipolaires) ramène les ions d'oxygène pour combler les lacunes du filament. Au HRS, le courant subsistant est beaucoup plus faible en raison de la dimension réduite du filament conducteur.

La figure 2.2 illustre le comportement des mémoires. Le mécanisme physique permettant de passer de l'état HRS à l'état LRS (communément appelé le SET) s'enclenche lorsqu'un champ électrique assez puissant détache les atomes d'oxygène de la couche active en les attirant en direction de l'anode (top électrode), laissant derrière eux des lacunes dans le matériau menant à la formation de filament(s) conducteur(s). C'est ce mécanisme qui est d'abord réalisé avec une tension particulièrement élevée pour "former" la mémoire en générant ses premières lacunes. Le mécanisme miroir RESET provoque le retour des ions d'oxygène de l'anode aux lacunes afin de les combler. Les mémoires résistives sont non-volatiles (elles maintiennent leur valeur de résistance même sans alimentation).

Les mémoires résistives comportent une variabilité inhérente attribuable principalement à l'échelle nanométrique de leur filament conducteur et à leurs conditions environnantes et leurs conditions de tests. Elles présentent notamment des non-idéalités affectant leur utilisation dont :

- des changements de comportement en fonction de la température [26]
- des problèmes de rétention (capacité de la mémoire à préserver son état de conductance au travers le temps)
- une programmation imparfaite des mémoires
- une perte de potentiel dûe à la résistance des électrodes
- du bruit (RTN et autre)
- une non-linéarité des lectures. Les mémoires ne suivent pas parfaitement la loi de Ohm.
- des échecs de mémoire (rendement non parfait des crossbars avec plusieurs mémoires coincées en état HRS ou LRS)

2.2.2 Programmation des mémoires

Une méthode d'écriture des mémoires résistives permettant des résistances variées entre le HRS et LRS est la programmation pulsée. En appliquant des trains de pulses aux caractéristiques variables (durée, amplitude de la tension, nombre de pulses et polarité), il est possible de faire varier relativement précisément la résistance des mémoires (augmentation de la résistance avec une polarité positive et diminution de celle-ci avec une polarité négative pour les mémoires bipolaires). L'augmentation de la résistance en résultant se nomme *Long Term Potentiation (LTP)* alors que sa diminution se nomme *Long Term Depression* (LTD). Un exemple de cette technique est présenté dans la figure 2.3. Des pulses sont appliqués sur la mémoire (en incrémentant l'amplitude) jusqu'à ce qu'elle atteigne la résistance voulue ; en cas de dépassement de la valeur, une boucle de rétroaction change la polarité et l'amplitude des pulses jusqu'à ce que la résistance converge. En pratique,

l'écriture des mémoires est limitée par leur quantité d'états distincts (finis) entre le HRS et le LRS [27] et dépend de la sensibilité individuelle de celles-ci.

2.3 Matrices de poids synaptiques sur crossbar de mémoires résistives

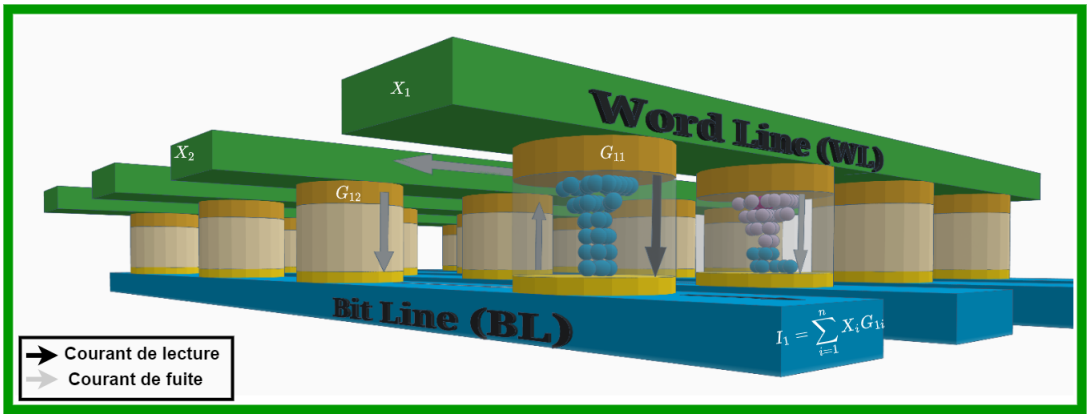

Figure 2.4 Illustration d'un crossbar de mémoires résistives. La valeur de la somme des courants selon la loi des noeuds de Kirchoff et la loi d'Ohm est illustré. L'empilement bottom électrode (en bleue) - mémoire (représenté par un cylindre) - top électrode (en vert) et du trajet des courants de fuite potentiels est également illustré.

Grâce à une utilisation de mémoires résistives en configuration crossbar (tel que représenté dans la figure 2.4), il est possible de garder en mémoire et d'opérer directement sur des matrices de poids synaptiques. Les rangées horizontales d'un crossbar sont appelées en anglais des *word lines* et les colonnes sont appelées des *bit lines*. En utilisant la loi d'Ohm de pair avec la loi des noeuds de Kirchhoff (KCL) après avoir appliqué des tensions d'entrées par les word lines, il est possible de calculé physiquement la somme pondérée représentée dans l'équation 2.1 à la sortie des bit lines. Ceci est fait en lisant le courant global ; la somme des tensions d'entrée du crossbar (X_i) multiplié par la conductance des mémoires résistives (G_{ij}, chacune modélisant un poids synaptique). La fonction d'activation est alors appliquée sur la somme par un circuit connexe ou par un traitement logiciel. Les architectures crossbar ont plusieurs configurations possibles : 1T1R [29] (un transistor est utilisé sur chaque mémoire résistive), 1S1R [30] (un matériau sélecteur est utilisé sur chaque mémoire résistive) et 0T1R [22]. D'autres architectures sont plausibles telles que l'architecture 1TNR (1 transistor pour N mémoires résistives composant une rangée). Les mémoires résistives fabriquées au 3IT pour ce projet sont de type 0T1R autrement connu sous le nom de crossbar passif. La précision des lectures et des écritures dans un crossbar passif est limitée par les courants de fuite. La figure 2.4 b) représente ce phénomène.

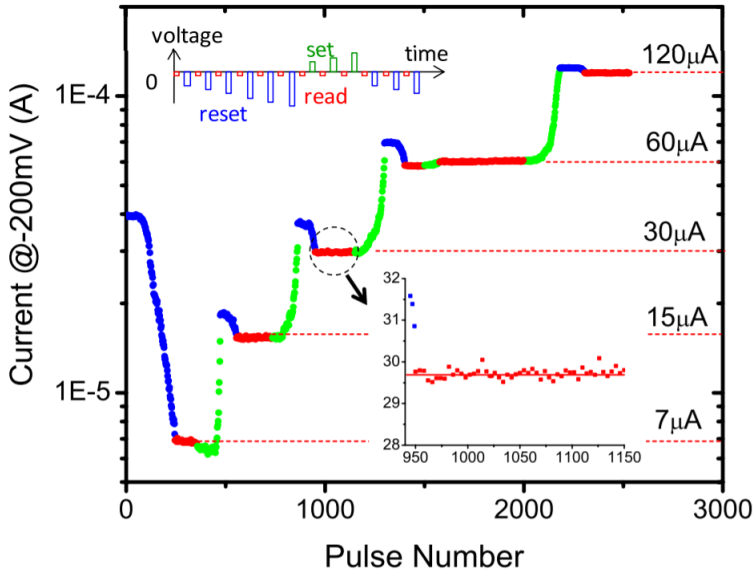

FIGURE 2.3 Reproduit de [28] © IOP Publishing. Reproduced with permission. All rights reserved. Programmation pulsée d'une mémoire résistive. Cinq états sont atteints : 28.5kΩ (7μA), 13.3kΩ (15μA), 6.7kΩ (30μA), 3.3kΩ (60μA) et 1.7kΩ (120μA). Chaque état de résistance est vérifié par 200 pulses de lectures de 200mv et demeure dans un écart acceptable de 1% de la valeur désirée.

Cette configuration vient cependant avec un défi de mise à l'échelle et une réduction de la consommation énergétique.

L'apprentissage machine sur crossbar de mémoires résistives peut se faire de façon *ex-situ* (déplacé) ou *in-situ* (sur place). L'approche ex-situ entraîne le réseau de manière purement logicielle (selon les approches conventionnelles) afin de déterminer la valeur numérique de ses paramètres et puis les associe à des valeurs de conductances équivalentes entre le HRS et le LRS des mémoires. La précision à laquelle cette association peut être réalisée dépend de la résolution de la gamme de conductances de la mémoire, de son nombre d'états stables et de sa précision d'écriture. Elles sont ensuite programmées pour atteindre les valeurs calculées. Cette approche ne fait qu'améliorer (en termes de vitesse et de consommation énergétique) l'inférence réalisée à l'aide du réseau sans influencer la performance de son entraînement. L'approche in-situ, quant à elle, réduit dramatiquement la dépendance du système au processeur en réalisant l'entraînement directement sur les mémoires résistives. Son implémentation est significativement plus complexe mais le fait de réaliser l'entraînement sur puce assure une conscience inhérente de la variabilité des mémoires.

2.4 Réseaux neuronaux stochastiques sur crossbar

Référence	Device Stack	Simulation ou expérimentale	Benchmarks	Imperfections considérées*	Méthode
[31]	ReRAM	Hybrid	MLP (MNIST)	B	Étude de l'impact sans compensation proposée
[32]	ReRAM	Simulation	VMM	PP	Ajout de résistances externes pour compenser l'erreur
[33]	ReRAM	Simulation	VMM	NL,PP	Étude de l'impact sans compensation proposée
[34]	ReRAM	Hybrid	MLP (MNIST)	B,PP	Moyenner la sortie de plusieurs réseaux ensemble
[35]	ReRAM	Hybrid	MLP (MNIST)	B	Étude de l'impact sans compensation proposée
[36]	ReRAM	Simulation	MLP (MNIST)	E	Adaptation des poids en fonction des problèmes spécifiques à chaque puce
[37]	ReRAM	Hybrid	MLP (MNIST)	NL,E,NLP,EP	Étude de l'impact sans compensation proposée
[38]	ReRAM	Hybrid	ResNet (ImageNet,CIFAR-10)	B,R,EP	**Entraînement conscient de la variabilité Calibration sur puce**
[39]	ReRAM	Simulation	MLP (MNIST) CNN (CIFAR-10)	E,EP	Calibration sur puce
[40]	ReRAM	Simulation	SLP (MNIST)	PP,EP	Calibration spécifique à la puce
[41]	ReRAM	Simulation	SLP (MNIST)	E,EP	Calibration sur puce
[42]	ReRAM	Hybrid	CNN (CIFAR10)	B	**Entraînement conscient de la variabilité**
[43]	ReRAM	Hybrid	VMM MLP(MNIST)	PP,NL	Conversion des poids algorithmique
[44]	ReRAM	Simulation	MLP(MNIST)	E	Réentraînement et distribution des poids sur le crossbar
[45]	ReRAM	Simulation	Lenet (MNIST) ResNet (CIFAR)	E	Étude de l'impact
[46]	ReRAM	Hybrid	MLP (MNIST)	E,EP	Calibration spécifique à la puce Compensation de la température
[47]	ReRAM	Hybrid	MLP (MNIST)	PP,NL	Optimisation de l'algorithme de programmation
[48]	eFlash, ReRAM	Hybrid	ResNet (ImageNet) Conv. DNN (CIFAR-10)	T,B,R,NL,EP,E	**Entraînement conscient de la variabilité Adaptation de l'algorithme de programmation**
Projet de maîtrise	ReRAM	Hybrid	SLP (Half-Moon)	B,R,EP,E	**Entraînement conscient de la variabilité**

Tableau 2.1 Tableau adapté de [48]. Tableau de comparaison des articles dans la littérature ayant des approches similaires à celle présentée dans le mémoire. . * : B : bruit, R : rétention, EP : erreur de programmation, E : échecs (mémoires coincées à un état résistif donné), NL : non-linéarité, PP : perte de potentiel, NLP : non-linéarité de la programmation.

Plusieurs articles dans la littérature proposent des solutions pour mitiger les pertes de précision dues à la variabilité des mémoires ou alors des interprétations de celles-ci qui furent particulièrement pertinentes à ce projet. Ces articles sont regroupés et comparés dans le tableau 2.1. La plupart des références énumérées dans ce tableau se concentrent sur une ou deux seules sources de variabilité à la fois. Parmi ceux-ci, certains présentent une étude de l'impact de la variabilité sans proposer de solution ou alors proposent des solutions partielles, spécifiques et moins polyvalentes que l'entraînement conscient de la variabilité présenté dans ce travail. Certaines de ces solutions sont d'ordre purement matérielle telle que la référence [32] dans laquelle des résistances sont ajoutées en série avec certaines électrodes des WL afin d'uniformiser la chute de tension parasitaire vue par les dispositifs. Quelques articles seulement utilisent une forme d'entraînement conscient de la variabilité pour converger vers une solution prenant en compte l'incertitude des mémoires. Ils seront adressés brièvement dans les prochains paragraphes :

Accurate deep neural network inference using computational phase-change memory [38] :

Afin ajouter du bruit aux poids synaptiques qui seront stockés dans la mémoire à changement de phase (PCM), les auteurs de l'article proposent d'ajouter le bruit aux poids lors de l'entraînement et de réaliser la rétropropagation des gradients par rapport aux poids sans y ajouter de la variabilité. La figure 2.5 tiré de l'article explique le processus

d'entraînement conscient de la variabilité. Fig. 2.5 a présente la structure du réseau alors que Fig. 2.5 b présente le processus d'entraînement. La sous-figure b est généralement applicable à tous les réseaux entraînés conscients de la variabilité. Ce groupe de recherche fut le premier à ajouter un bruit corrélé à la variabilité de leurs dispositifs aux poids à l'aide de l'astuce de reparamétrisation. L'astuce de reparamétrisation est une technique utile pour l'apprentissage machine comportant une composante stochastique au niveau des poids. Elle est utilisée dans plusieurs architectures de réseaux de neurones, dont les *variational auto-encoders* (VAE) ou dans les réseaux bayésiens. L'utilisation de cette technique permet de rendre la sortie d'un réseau *différentiable* par rapport à ces entrées. Il est sinon impossible de réaliser la rétropropagation des poids au travers un noeud stochastique (*stochastic node*).

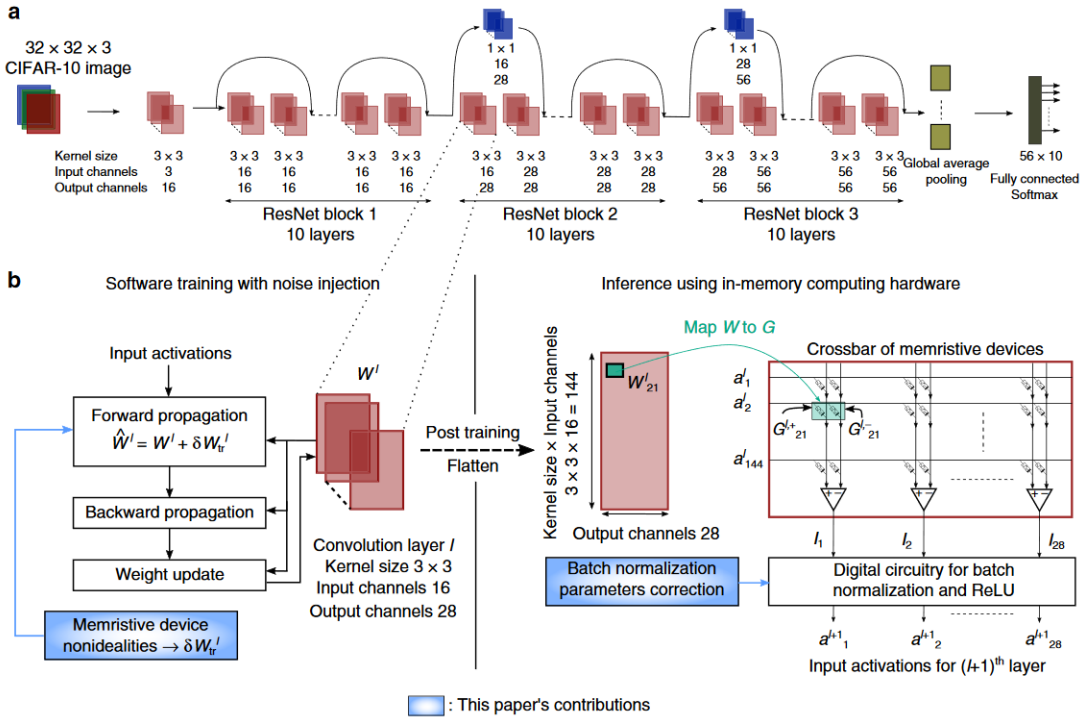

FIGURE 2.5 Reproduit de [38] sous license CC BY 4.0. Schéma représentant un exemple d'entraînement conscient de la variabilité pour le jeu de données CIFAR-10.**a** la structure du réseau prenant en entrée les images de CIFAR-10. **b.** (gauche) l'entraînement conscient de la variabilité dans lequel du bruit est ajouté aux poids lors de la passe avant du réseau, c'est dans cette étape que prend place la reparamétrisation des poids. (droite) La procédure après entraînement pour transformer les poids synaptiques en conductances d'un crossbar.

Le DNN est finalement entraîné en utilisant du *dropout*, une technique de régularisation couramment utilisée en apprentissage machine afin d'améliorer la généralisation des

modèles en désactivant aléatoirement un certain pourcentage des neurones d'une couche cachée lors d'une passe avant du réseau. Les sources de variabilité qu'ils considèrent sont l'erreur de programmation, la rétention et les échecs de mémoires. Après avoir programmé les poids synaptiques sur 723 444 PCMs d'une puce, la précision du réseau est restée au-dessus de 92.6% après une journée entière (en simulation). Ceci fit de ce réseau (au moment de sa publication) le réseau le plus précis démontré sur le CIFAR-10 dataset implémenté sur mémoires résistives analogues (au dire des auteurs).

Bayesian neural network enhancing reliability against conductance drift for memristor neural networks [49] :
Ce réseau réalise une forme de réentraînement conscient de la variabilité sur crossbar 1T1R. Il se distingue de ses semblables du fait qu'il considère un décalage de la conductance des mémoires à travers le temps (*conductance drift*) et une approche bayésienne pour «optimiser »les poids synaptiques en fonction de celui-ci. Le réseau neuronal utilisé est en fait un ensemble de petits réseaux déterministes programmables sur un seul crossbar. Un réseau neuronal classique est d'abord entraîné par une rétropropagation des gradients pour obtenir les valeurs de poids optimales : w_i. Un BNN est alors initialisé avec une distribution à priori sur les poids $p(w)$ correspondant à un ensemble de distributions normales dont les valeurs de poids moyennes μ_i équivalent à w_i ($p(w) = N(w_i, \sigma_i^2)$). La valeur de σ est l'écart-type associé à un décalage acceptable et réaliste des conductances et doit être déterminée expérimentalement sur les mémoires. L'article stipule que cette association des distributions à priori aux résultats d'un réseau déterministe accélère considérablement la convergence (ce qui est assez intuitif). Après l'entraînement du réseau, les poids synaptiques de l'ensemble sont programmés à des valeurs de μ échantillonnés depuis ses distributions et la véritable conductance atteinte variera selon le décalage de conductance, l'imprécision d'écriture ainsi que d'autres facteurs. Cette imprécision lui assure d'être approximativement échantillonnée depuis la distribution. Lors de l'inférence, la moyenne des sorties des réseaux de l'ensemble est considérée comme étant la solution, ce qui est fonctionnellement identique à l'équation B.4 de la section de l'annexe B.0.1. Pour rendre compatible BbyB avec la distribution à priori tel que définie dans l'article, la fonction de la distribution à priori de l'algorithme (voir eq. B.10 de la section B.0.1) a dû été utilisée avec $\pi == 0||1$. La configuration 1T1R considérée les affranchit des courants de fuite et des effets de méthode d'adressage (V/2 ou V/3) des crossbars 0T1R. l'apprentissage machine adaptée aux non-idéalités des ReRAM des modèles 1T1R sont généralement peu applicables aux dispositifs fabriqués au 3IT à cause de l'absence de l'effet des schémas de biais.

Mitigating Imperfections in Mixed-Signal Neuromorphic Circuits [48] et Improving noise tolerance of mixed-signal neural networks [40] :
L'article *Mitigating Imperfections in Mixed-Signal Neuromorphic Circuits [48]* est une continuation de l'article *Improving noise tolerance of mixed-signal neural networks [40]* réalisé par le même groupe. La figure 2.6 illustre les différences entre les types d'entraînement prenant plus ou moins conscience des variabilités et énumère celles prises en compte par le réseau hardware-aware considéré dans l'article en question. Cet article amène comme principale nouveauté par rapport à l'article [38] l'intégration de plusieurs nouvelles sources de variabilité dont la température, la non-linéarité des lectures et les échecs de certaines mémoires. Les performances recensées dans l'article sont impressionnantes avec des améliorations énergétiques de 2.5x à 9x par rapport à un réseau régulier durant l'inférence et des pertes de précisions de moins de 1% en considérant des températures entre 25 et 100 degrés Celsius. La tolérance aux défauts est améliorée de 100x et le tout est démontré sur un crossbar passif de 64x64 mémoires avec 25% de variations aux mécanismes de permutation des mémoires en simulation.

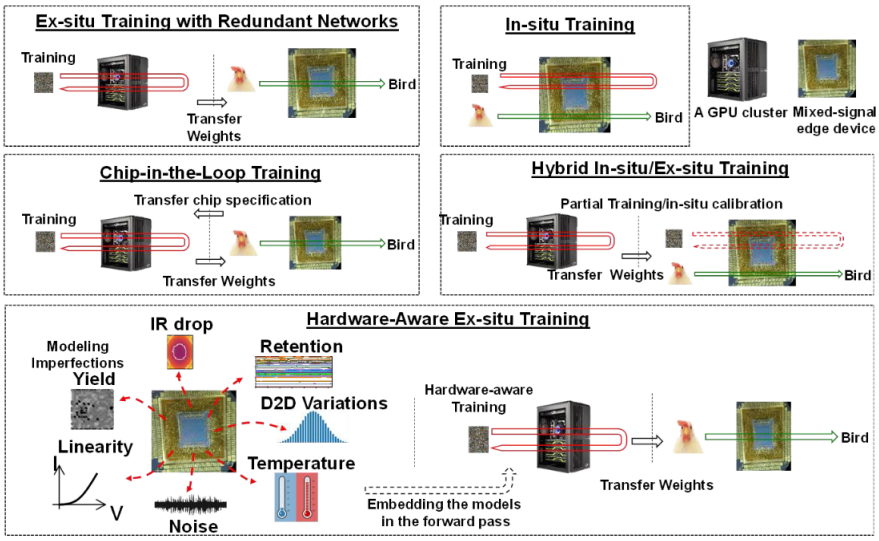

Figure 2.6 Reproduit de [48]. Représentation schématique des différents types d'entraînement et de transfert subséquent sur crossbar de mémoires resistives. (haut) L'entraîment ex-situ est comparé à l'entraînement in-situ. (milieu) L'entraînement de type chip-in-the-loop dans lequel certaines caractéristiques d'une puce précise sont prises en compte ainsi que l'entraînement Hybrid In-situ/Ex-situ qui réalise d'abord un entraînement ex-situ en logiciel et puis transfert ensuite les poids sur chip afin de calibrer le réseau aux variabilités matérielles. (en-bas) Les variabilités prises en compte dans l'entraînement hardware-aware de l'article

Le projet de maîtrise propose une alternative plus empirique à cet article pour entraîner le réseau neuronal stochastique de manière *hardware-aware*. Ce groupe utilise cependant la méthode d'adressage V/2 au lieu de V/3 et emploie des modèles de variabilité plus complexes et moins empiriques que les nôtres. Les différences plus notables entre ces deux articles sont énumérées dans l'article suivant cette section.

CHAPITRE 3

Article

3.1 Avant-Propos

3.1.1 Auteurs et affiliations

Philippe Drolet :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
Pierre-Antoine Mouny :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
Matthieu Valdenaire :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
Raphael Dawant :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
Javier Arias Zapata :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
Pierre Gliech :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
Yann Beilliard :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
Fabien Alibart :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada et Institute of Electronics, Microelectronics and Nanotechnology (IEMN), Université de Lille, 59650 Villeneuve d'Ascq, France
Serge Ecoffey :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
SEAN U. N. Wood :Department of Electrical and Computer Engineering Université de Sherbrooke Sherbrooke, QC, Canada
Dominique Drouin :Institut Interdisciplinaire d'Innovation Technologique (3IT), groupe INPAQT, Université de Sherbrooke, Sherbrooke, Québec J1K 0A5, Canada
Date d'acceptation : à venir
État de l'acceptation : n/a
Revue : Advanced Intelligent Systems
Référence : arXiv:2305.18495

Titre français : Entraînement conscient de la variabilité améliorant l'inférence de réseaux neuronaux analogues sur crossbar passif de mémoires résistives de TiO_2

Contribution au document : L'article récapitule le processus de caractérisation ayant permis l'acquisition des données pour les modèles de variabilité, explique la structure et le fonctionnement des réseaux neuronaux conscient et inconscient de la variabilité et présente les résultats principaux du projet de maîtrise. L'ensemble des objectifs et des sous-objectifs précédemment énumérés y sont donc abordés.

Contribution de l'auteur à l'article : L'auteur du mémoire a effectué la plupart des mesures, a conçu le système de test (logiciel et matériel), a conçu et entraîné les réseaux neuronaux et a analysé les données acquises. L'auteur du mémoire a également écrit l'article avec des commentaires et des contributions de tous les autres auteurs.

Résumé français : Les crossbars passifs de mémoires résistives, une technologie prometteuse pour les multiplications vecteurs-matrices réalisées de façon analogique, sont supérieurs à leur configuration soeur 1T1R en termes de densité d'intégration. Cependant, les transferts de poids synaptiques en conductances équivalentes sur crossbar passif sont présentement accompagnés de pertes de précision importantes. Ces pertes sont dues aux imprécisions matérielles telles que les courants serpentins, les changements de conductance due au schéma de biais et aux erreurs de programmation des mémoires. Dans cet article, nous proposons de nouvelles approches d'entraînement adaptant les techniques d'apprentissages machines habituelles (tel que le dropout, l'astuce de reparamétrisation et la régularisation) aux variabilités mesurées des crossbars de TiO_2 afin de générer des modèles se prêtant mieux au transfert sur crossbar de mémoires résistives. La viabilité de cette approche est démontrée en comparant les sorties et la précision de deux réseaux neuronaux, dont un classique et un conscient de la variabilité, à l'aide d'un jeu de données de demi-lunes dans une simulation fondée sur des données expérimentales. En entraînant le réseau en utilisant la méthode proposée, 79,5% des données du jeu de données de test sont classifiées avec 95% de précision, comparés à seulement 18,5% des données pour le réseau inconscient de la variabilité.

Note : À la suite des corrections qui seront proposées par les évaluateurs, cette version de l'article ne sera pas la même que celle qui sera publiée dans un journal. Cette version incorpore certains nouveaux commentaires et diffère donc de la version publier sur l'arXiv.

3.2 Manuscrit

Hardware-aware Training Techniques for Improving Robustness of Ex-Situ Neural Network Transfer onto Passive TiO_2 Re-RAM Crossbars

3.2.1 Abstract

Passive resistive random access memory (ReRAM) crossbar arrays, a promising emerging technology used for analog matrix-vector multiplications, are far superior to their active (1T1R) counterparts in terms of the integration density. However, current transfers of neural network weights into the conductance state of the memory devices in the crossbar architecture are accompanied by significant losses in precision due to hardware variabilities such as sneak path currents, biasing scheme effects and conductance tuning imprecision. In this work, training approaches that adapt techniques such as dropout, the reparametrization trick and regularization to TiO_2 crossbar variabilities are proposed in order to generate models that are better adapted to their hardware transfers. The viability of this approach is demonstrated by comparing the outputs and precision of the proposed hardware-aware network with those of a regular fully connected network over a few thousand weight transfers using the half moons dataset in a simulation based on experimental data. For the neural network trained using the proposed hardware-aware method, 79.5% of the test set's data points can be classified with an accuracy of 95% or higher, while only 18.5% of the test set's data points can be classified with this accuracy by the regularly trained neural network.

3.2.2 Introduction

As neural networks' complex problem-solving capabilities increase, so do their energetic and computational demands. These demands have grown so much that the use of graphics processing units (GPUs) and data centers is proving to be essential for practical machine learning. Meanwhile, a plethora of applications at the edge would greatly benefit from the use of neural networks but cannot always use these deep networks due to their high energy demands [50]. As the main limitations of neural network computations stem from the von Neumann bottleneck between the memory and processor [51], in-memory computing with analog, non-volatile emerging resistive memories is among the most promising avenues of innovation that can be used to overcome this problem [52, 53, 54]. Non-volatile memories exploit metal oxide [55, 56, 57, 58], phase-change [59] or ferroelectric materials [60], among other things, to perform computations directly in memory, decreasing the need to fetch

data to perform computations. In a crossbar configuration, in-memory computing relies on the fundamental Ohm's and Kirchoff's laws, which enable an energy-efficient version of the vector matrix multiplications (VMMs) at the core of neural networks. These memory devices are versatile and have proven their potential for many different types of neural networks, such as long short-term memory (LSTMs) [61, 62], generative adversarial networks (GAN) [63] and Bayesian NNs [64, 49, 65]. Neural networks on crossbars always require either in-situ or ex-situ learning techniques. Both of which have been demonstrated on TiO_2 crossbars [66]. The former technique trains the network online on the devices by implementing backpropagation through tuning pulses. The latter performs the training offline and thereafter the network is transferred to the crossbars by converting the obtained trained weights into conductances. The more commonly used ex-situ technique is much simpler but hardly considers hardware variabilities during training. This renders it less robust to all potential hardware non-idealities. The crossbar architectures presented in the literature are often based on the 1T1R integration scheme [67, 68], where a transistor (1T) is cascaded with each memory (1R) to allow independent accesses to desired memories without disturbing adjacent devices. This improved control provides high programming and reading accuracies but leads to a lower integration density, higher wiring complexity and increased fabrication costs. On the other hand, passive (0T1R) crossbars exhibit excellent scalability and lower fabrication costs. In [69], a factor of 20 between the sizes of individual 1T1R (4 μm) and 0T1R (200 nm) memristors was reported. An average relative conductance import accuracy of 1% on a fully passive TiO_2 crossbar with a size of 64×64 was reported [57]. This crossbar had $\sim$ 99% of memristors functioning. As such, passive TiO_2 crossbars hold much promise for the future of in-memory computing with ReRAM devices.

However, these passive crossbars present multiple inherent sources of variability [70], which currently hinder their use at large scale. The conductance tuning imprecision that is observed when a device is programmed to a certain conductive level, unsatisfactory device yields, asymmetrical switching dynamics, the conductance drift due to relaxation effects over time and sneak path currents are a few examples of the issues afflicting this technology.These non-idealities are thoroughly explained in [3] and are common to most memristor technologies. They directly affect the results of analog matrix-vector multiplications (VMM). Cumulatively, they often lead to incorrect classifications. While some of these non-idealities can be harnessed to implement efficient stochastic neural network models in hardware [71, 56, 65, 61], they remain very detrimental for standard in-memory computing solutions and make it difficult to scale up this technology. A potential avenue that can be used to mitigate the sneak path current problems is the use of tiled crossbars ; the neural

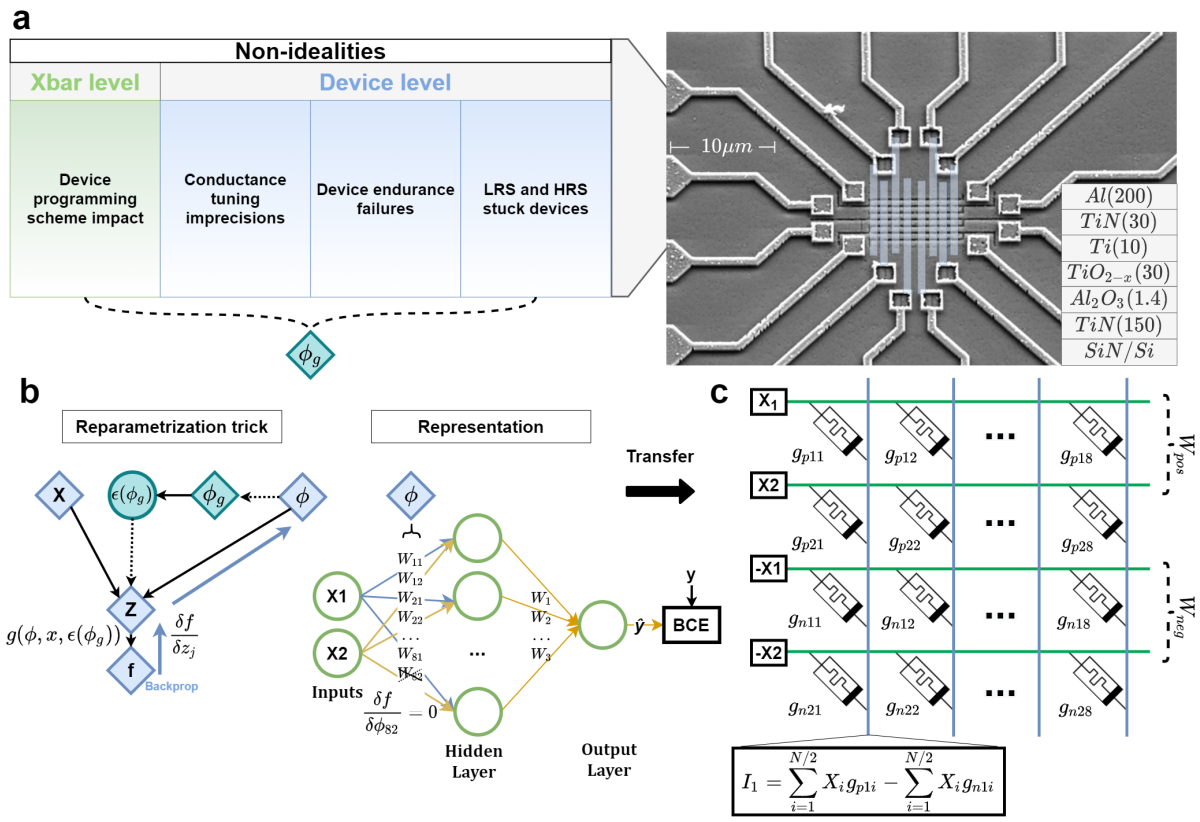

FIGURE 3.1 Schematic illustration of the main concept investigated in this work. a. (Right) A scanning electron microscopy (SEM) picture of our 8×8 TiO_2 crossbar architecture. The top electrodes (word lines) are visible, while the bottom electrodes (bit lines) fabricated by a damascene process have been represented in blue. The materials stack of the resistive memory devices is displayed in the inset. (Left) The different sources of variability considered during training are shown. **b.** Depiction of the training process of the neural network. The left side of the sub-figure represents the reparametrization trick in relation to the weight values. The right side is a schematic representation of the neural network investigated in the scope of a half moon toy problem.. A dotted line indicates a conversion between ranges (weight range to conductance range or vice versa). The weights are initially converted to conductances with differential weights ($G_+ - G_-$). A noise parameter ϵ is then added to both the positive and negative devices through a reparametrization trick. A percentage of neurons, depicted in the schematic by a half-dashed X in the hidden layer for weight W_{82}, are, similarly to dropout, deactivated by conversion to a failure state in the low resistance state (LRS) or high resistance state (HRS) (instead of by being set to 0 as they would be in the usual dropout method). The gradient on these neurons is then zeroed out during backpropagation. **c.** Physical implementation on passive crossbar array. The weights are represented by two devices to be able to represent signed values.

network weights can be separated onto sub-crossbars called tiles. Tiles are compatible with control transistors and circuitry. These 8×8 crossbar tiles offer an intermediate solution

that lies between the improved controllability of 1T1R crossbars and the scalability of 0T1R crossbars.

In this work, we report a novel hardware-aware ex-situ training approach that improves the robustness of neural network models against the non-idealities of TiO_2-based memory crossbars. The main contributions of this study are the introduction of new, fully data-driven and computationally simple characterization and variability modeling methods that are used jointly with a reparametrization trick during training to mitigate precision losses when weights are converted into their equivalent ReRAM conductances on a passive crossbar. This procedure is often referred to as hardware-aware training. The simple variability models do not require computationally prohibitive mathematical and physical models, and they lead to good performance without requiring multiple rounds of tuning ; these models require only reproducible characterizations that are all easy to automate. Figure 3.1 shows the overall hardware-aware procedure. The hardware-aware network provides accurate classification results for the half moons dataset for a much greater percentage of the simulated transfers compared to a regular neural network. This greater accuracy across transfers proves that the hardware-aware procedure is robust to significant biasing-scheme-effect conductance changes (up to 60 μS), conductance tuning imprecision and neural network weights impacted by random substitutions to account for stuck devices. The devices studied in this work are 0T1R devices, and they can be seen in Figure 3.1 a and in Supplementary Figure A.1.

Other works, such as [49, 72, 73], employ a similar approach. In [49], an impressive performance was reported but the authors chose not to consider device failures and only consider 1T1R devices, therefore neglecting biasing scheme effects. Meanwhile, [72, 73] deal with 0T1R TiO_2 passive crossbars and implement many different variability sources such as temperature, biasing scheme effect and non-linearity in their hardware simulations and training to great effect. However, the relationship between the variability and the crossbar position of the devices is not considered in these works. The focus is instead placed on a conductance tuning technique that mitigates biasing scheme disturbances. Also, the fact that our method depends solely on empirical data means that it is also applicable to different memristive technologies with slight modifications, rendering it highly adaptable. The biasing scheme considered in these other works is the V/2 scheme ; the V/3 scheme is considered in this work and the variability modeling approaches are therefore different. The method proposed only requires an initial round of device characterization for a new memristive technology. It is therefore applicable to any memristor technology

with a similar conductance tuning procedure and the hardware overhead of this initial characterization remains advantageous over 1T1R topologies.

In this paper, we begin by explaining the different sources of variability that are taken into account by the hardware-aware network, along with the characterization process that led to the creation of each of these sources. We continue with a software demonstration that compares the performances of weights trained considering these variabilities and weights trained naively over 10,000 simulated ex-situ weight transfers for a simple binary classification problem. The hardware-aware network significantly outperforms the regular neural network.

3.2.3 Results and Discussion

As previously mentioned, passive crossbars are afflicted by many different and often indirectly correlated non-idealities. The device-to-device and crossbar-to-crossbar variabilities are considerable and difficult to predict. Depending on a specific crossbar's properties, a transferred neural network may perform well or very poorly. The proposed approach aims at using neural networks' learning abilities to learn how to classify data points in spite of the effects that non-idealities will have on analog VMMs. These non-idealities were estimated through the extensive electrical characterization of TiO_2-based resistive memories fabricated in 8×8 crossbar and crosspoint configurations. The considered non-idealities were those deemed to have the greatest effect on VMMs, namely the conductance tuning imprecision, biasing scheme effects and high resistance state (HRS) and low resistance state (LRS) stuck devices.

During training, the neural network weights are converted from their software range to the predetermined mean achievable conductance range G_R for every batch of data. This conductance range GR is chosen as the narrowest measured conductance range of working devices. The weights are then shifted according to randomized variability samples using a reparametrization trick [74] integrated into a modified version of the PyTorch library [11]. The use of the reparametrization trick allows the Monte Carlo estimate of the expectation (every time we sample new weights) to be differentiable with respect to the weight parameters and enables backpropagation, as depicted in Figure 3.1b. The process is explained by the flowchart shown in Figure 3.1 and the mathematical details are given in Supplementary Note 1. All considered non-idealities are numerically added into a matrix ϕ_g created from the conversion of the network's parameters ϕ to the conductance range G_R and share the same dimensions. This matrix is converted back into the weight range after variabilities are added. The now-converted matrix is subtracted from the original parameter matrix ϕ to obtain the $\epsilon(\phi_g)$ matrix used in the reparametrization trick. G_R was determined to

be from 100 μS to 400 μS by electrical characterizations made on our 8×8 crossbars [55]. The randomized variability samples depend on the conductance tuning imprecision and the biasing scheme effects recorded in a database. Percentages of neural network weights x and y corresponding to LRS and HRS stuck devices are also replaced by equivalent high or low conductance values. This effectively simulates the result of an ex-situ transfer on a crossbar, as depicted in Figure 3.1c, with randomized programming failures for every batch of training data. It also ensures that the network considers randomized potential sources of hardware-related errors. The trained network will converge towards the solution that is most robust to the average of all potential crossbars, considering our TiO_2 memristive technology non-idealities.

The tests performed to gather the data in the non-ideality database used during training for tuning imprecision, biasing scheme effects and finally HRS or LRS stuck devices are described in the following subsections. In the case of the weight disturbances induced by the biasing scheme effect, fitting normal distributions to the data was unfortunately impossible. The Shapiro-Wilk tests for normality of the distributions fitted to the data usually returned an alpha value of less than 0.05, causing the null hypothesis to be rejected [75]. A sample from this distribution does not approximate a recorded disturbance instance well. The approach taken instead was to sample a recorded conductance change from a portion of the database consisting of only raw data. This is different from sampling from a fitted distribution, which is done for the conductance tuning imprecision. In all of the tests, the conductance tuning procedure followed the pulse tuning technique presented in [76] with a 1% tuning tolerance.

Device conductance tuning imprecision

As shown in Figure 3.1a, the device-level conductance tuning imprecision is considered in the training process. The studies undertaken to characterize it were realized by dividing the achievable resistive ranges of all 29 studied devices into eight different equidistant levels and repeatedly performing a tuning/untuning procedure. Figure 3.2 illustrates how the statistical database was established, along with an example. Figure 3.2a shows the experimental data behind one instance of the procedure for a target conductance of 125 μS. The distribution of read values after the conductance reached and stabilized around the target conductance is shown in Figure 3.2b. This tuning/untuning procedure was repeated 40 times per level and per device. In Figure 3.2c and d, the data points corresponding to

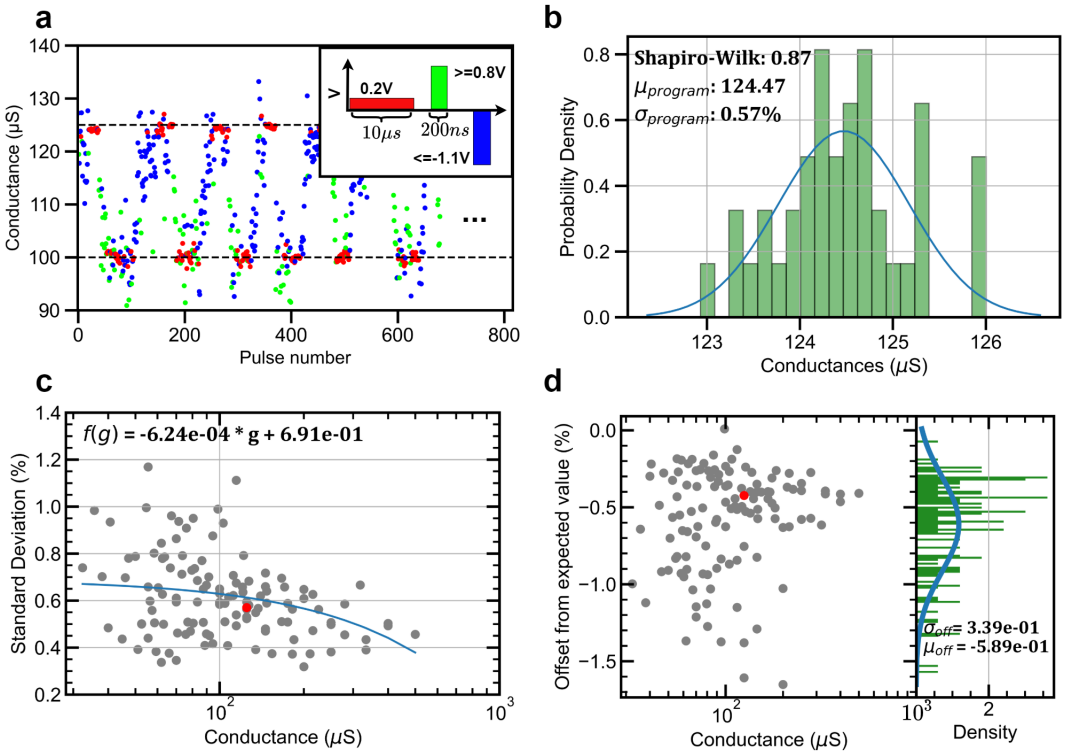

FIGURE 3.2 Experimental measurements of the conductance tuning variability of the TiO2-based resistive memories. a. The conductance is tuned from a low conductance state to the target conductance state using SET (in blue) and RESET (in green) pulses forty times. The last read value is then kept in memory so that **b.** a normal distribution may be fitted to the programmed conductances. A conductance value of 125 μS is targeted, starting from 100 μS in this example. A simple Shapiro-Wilk test is performed to see if the null hypothesis must be rejected. A mean and a standard deviation are extracted from this distribution. These two steps were repeated eight times across the conductive range of each individual device. A total of 29 devices were tested with this protocol. **c.** Depiction of the relationship between all of the standard deviations of the fitted normal distributions presented in panel **b** and the conductive level of the device. The average standard deviation increases as the conductance decreases, as predicted. The fitted linear relationship between the conductance and standard deviation is expressed on the plot. **d.** Depiction of the relationship of the offset with respect to the mean of the normal distributions, similar to the one shown in panel **b**, and their targeted (ideal) conductance with respect to the conductive level of the device. There is no clear correlation between the offset from the target value and the conductive level of the device. A normal distribution is instead fitted to the offsets.

the data distribution in Figure 3.2b are depicted in red and are evaluated as follows :

$$\begin{aligned}(x_c, y_c) &= (g_{target}, \sigma_{program}) = (125\mu S, 0.57\%)\,, \\ (x_d, y_d) &= \left(g_{target}, \frac{\mu_{program} - g_{target}}{g_{target}} * 100\right) \\ &= (125\ \mu S, -0.424\%)\,.\end{aligned} \tag{3.1}$$

Figure 3.2c shows the standard deviation of the programing precision as a function of the conductance state of the devices tested, where the values are expressed in terms of a percentage of the desired conductance value, while Figure 3.2d shows the offset between the mean of the read conductance values after the tuning of the devices and their target conductance values. Since the devices are always tuned from a low conductive level to a high conductive level, the mean offset of the achieved conductance from the ideal target conductance is always negative. Therefore, the transfer on a crossbar should be performed after all devices are RESET in order to match this replicable offset, meaning that all devices should start from the off state.

During training, when the network weights are transformed into software conductances g_{s+} and g_{s-} for signed weights or g_s for unsigned weights, a weight is sampled from a normal distribution centered around these values with a standard deviation that depends on the conductive level of the transformed weight. Figure 3.2c shows the function $f(g)$ defining this standard deviation relationship. This allows the network to take into account the correlation that is often observed between the conductive level and variability [56]. An offset is then sampled from a distribution $N(\mu_{off}, \sigma_{off})$, which is depicted in Figure 3.2d, and added to the sampled conductance value. Because the test is performed by reading ten times before considering a memory to be programmed, the read variability on devices is also taken into account. An experiment was undertaken to isolate the read variability of the characterization equipment with respect to the variability of the measured values ; it was found that the reading variability stemming from the instrument is one order of magnitude lower than that of the experimental data (Supplementary Figure A.5.)

Conductance disturbances caused by biasing scheme effect

Programming devices on a passive crossbar require the use of a biasing scheme to reduce sneak path currents. The V/3 scheme was used in this study since all devices other than the addressed memory will see the same voltage of V/3. This standardizes and simplifies the process of adding variability to the weights. It also makes it possible to use higher-amplitude voltages to tune the devices without affecting neighboring devices significantly. In order to measure the change in the conductance of a device due to the programming of its neighbors, we tuned devices from three different crossbars, for a total of 144 devices (64 + 64 + 16), to random values within the expected conductance ranges G_R. The

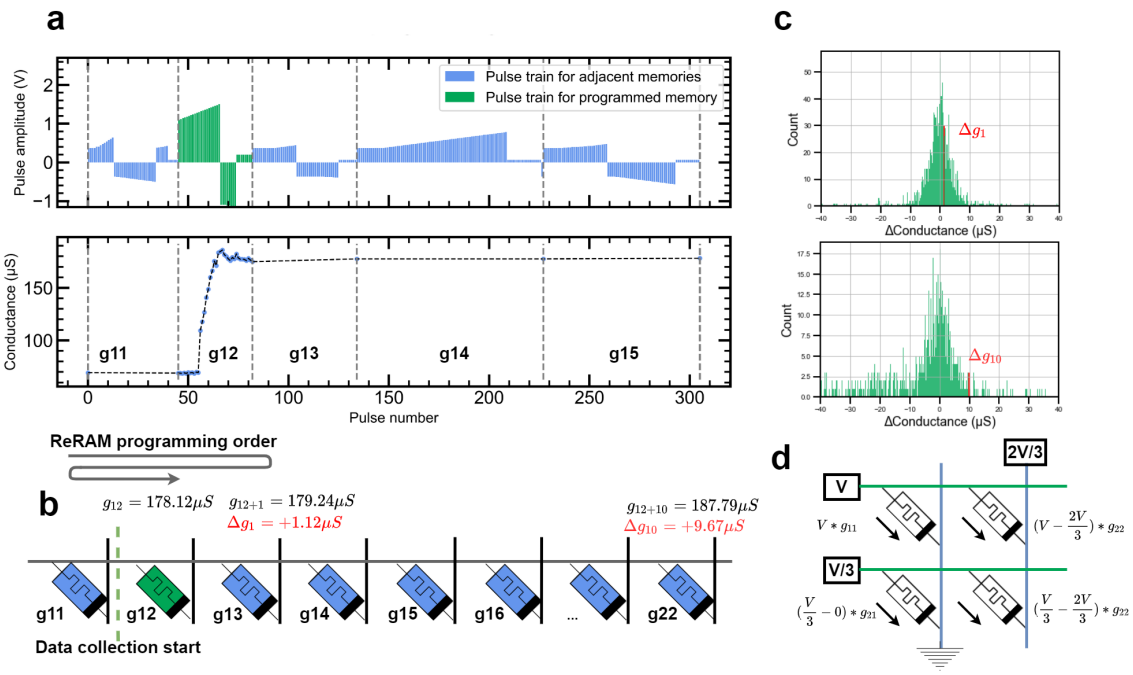

FIGURE 3.3 Depiction of the procedure for gathering data on device conductance changes caused by the biasing scheme. a. The top graph depicts the pulse train seen by a device during its own programming (in green) and during the programming of its neighbors (in blue). The bottom graph shows the conductance evolution of a device following the pulse trains displayed in the top graph. The device changes the conductance significantly during its programming cycle and changes it slightly during its neighbors' programming cycle. **b.** Color-coded depiction of a section of the crossbar studied in panel **a** with numerical values for the measured conductance change seen by the device after the programming cycle of other memories. **c.** Histogram of the recorded conductance changes with the bins containing the changes illustrated in panel **b** highlighted in red. The top graph contains the conductance changes when one additional device is programmed and the bottom graph contains the conductance changes when ten additional devices are programmed. **d.** Depiction of the V/3 and 2V/3 biasing schemes for programming devices.

disturbance of a device, after the tuning cycles for devices that are tuned after itself are complete, is recorded in the database. This disturbance is due to the V/3 biasing scheme. A thorough example is displayed in Figure 3.3a and b. The conductance changes caused by the biasing scheme accumulate through the crossbar programming, which can be observed by comparing the disturbance due to the programming of the neighbors of the device that is tuned first with that of the devices that are tuned last. As such, the database of recorded disturbance measurements is separated into sub-databases for each number of subsequent devices tuned after the programming of the current device. Figure 3.3c shows two such sub-databases, one for the scenario in which one additional device is tuned and one for

the scenario in which ten additional devices are tuned. The device that is tuned last is the bottom-rightmost weight of our neural network matrix when it is transferred from top to bottom and left to right. It will see no conductance change due to the biasing scheme since it is the last-tuned device. The devices stuck in an HRS or LRS or that could not reach their target conductances were also not added in this database. If they had been included, the database would have been polluted by stuck or ineffective devices, which would have biased it towards no conductance change.
As can be seen in Figure 3.3c, a tail of recorded crosstalk effects is more prevalent for negative values. This is an indirect consequence of the asymmetrical device switching mechanism between the SET and RESET pulses. More negative pulses are required to reset devices than positive pulses. Thus, devices will, on average, see more successive negative pulses than positive pulses. This phenomenon is worsened by the fact that devices are individually affected differently by negative pulses, with more sensitive devices leading to greater conductance changes. A conductance change of up to 60 μS was allowed in the sub-databases ; this change is six times larger than the lowest possible conductance weight. The cases in which the disturbance is greater than this value usually correspond to an HRS or LRS failure.

During the training of the neural network, a random offset sampled from these sub-databases correlated to the positions of the weights in the crossbar is added to each weight. The use of the V/3 scheme, as shown in Figure 3.3d, makes it possible to proceed in this way. This ensures that the first weights in the matrices will see more V/3 amplitude pulses and will, on average, have a greater variability than the last devices that are programmed.

HRS-LRS stuck device substitution

Some devices are stuck in a low or high resistive state due to defects. Regardless of the DC voltage or the pulse amplitudes that are applied to such devices, their conductive levels will never be tunable. As shown in Supplementary Figure A.4, LRS failures include retention failures. Such failures are caused by temporal effects and cause the device conductance to drift over time to a typically higher conductance state. HRS failures may occur for devices that were not forming free or devices with a very narrow achievable conductance range lying outside the desired range. To account for these device faults, percentages of devices x and y corresponding to HRS and LRS stuck devices, respectively, have both positive (g_{s+}) and negative (g_{s-}) software weight components that are replaced by a randomly sampled failed device conductance. The percentages x and y are fixed throughout the training, but the affected weights are randomized for every batch of input

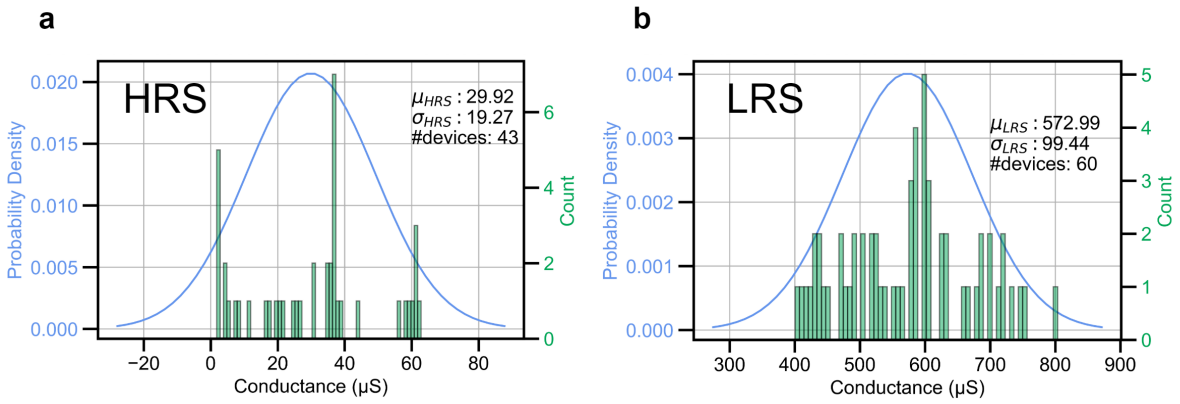

FIGURE 3.4 Distribution of faulty ReRAM. a. Recorded conductance levels of dead crosspoints stuck in an HRS. The Shapiro-Wilk p-value is 0.006 and therefore the hypothesis of the normality of the distribution is rejected. **b.** Recorded conductance levels of dead crosspoints stuck in an LRS. A device is considered stuck when it fails to change its conductance state after the application of 75 successive pulses of maximum SET or RESET amplitudes (2.0 V or -2.4 V).

data. This implies that certain weights will be doubly affected by this substitution if their positive and negative components are both assigned as failures.

The stuck devices' conductances are sampled from two distributions based on the characterization of LRS and HRS stuck devices made on five 8×8 crossbars (320 devices). The conductance range of devices stuck in an HRS is much smaller than that of devices stuck in an LRS ; HRS devices are defined as devices stuck below our HRS resistance of 100μS. The distribution of failed HRS state devices can be seen in Figure 3.4a, while the distribution of failed LRS devices can be seen in Figure 3.4b. Since fitting the normal distribution is inappropriate for the HRS distribution, as it fails the Shapiro-Wilk normality test, HRS stuck device faults are uniformly sampled from the range [100 μS, 10 μS]. The impact of devices stuck below 10 μS will be more negligible than the impact of those stuck at $<$100 μS and $>$10 μS. Indeed, devices stuck in a higher conductance state will have a greater impact on the summation of the currents during the analog matrix multiplication. Gradient descent is deactivated for the weights affected by HRS or LRS stuck weights. Typically, the fabrication yields of working devices on crossbars for state-of-the-art memristive devices are above 95% [77], with some articles reporting near perfect crossbars with yields of 99% [57] or even 100% [78]. The percentages of HRS and LRS stuck devices should be chosen to reflect this fact. In fact, in general, the values of these parameters should be chosen as conservatively as possible so that the network converges to a solution that is robust to as many device failures as possible. Of course, replacing too many weights with random values at the extremities of the conductance range and therefore at the extremities

of the overall weight range will prevent the network from converging to a solution, making the training impossible. This makes these percentage values tunable hyperparameters.

Mathematical expression

Overall, the mathematical equation representing the sampling of each of the devices for either positive or negative weight components is equation 3.2 :

$$\begin{aligned} & N\left(\mu_{w\pm}, {\sigma_{w\pm}}^2\right) \\ & = N\left(g_{s\pm}, f(g_{s\pm})^2\right) + N\left(\mu_{off}, \sigma_{off}^2\right) + g_{adj}(n_d), \\ & = N\left(g_{s\pm} + \mu_{off} + g_{adj}(n_d), \sigma_{off}^2 + f(g_{s\pm})^2\right), \end{aligned} \tag{3.2}$$

where g_s is the direct conductance equivalent of the weight after a linear transformation between the two ranges (see Supplementary Note 1). The weight's range is reevaluated for every batch during training since the maximum and minimum weight values are subject to change as the training progresses. $N\left(\mu_{off}, \sigma_{off}\right)$ is the sampled conductance tuning imprecision offset and $g_{adj}(n_d)$ is the sampled biasing scheme effect from programming adjacent devices with respect to the number of additional devices n_d programmed after the current device. The variability associated with a device in training is dependent on both its target conductance level for both negative and positive weight components and the position of the devices in their respective crossbar. Every weight matrix is presumed to be programmed on a tiled crossbar or on a sub-section of a crossbar.

The network learns to expect more variability from the first devices that are programmed because of the biasing scheme effects and from devices at low conductance values and will adjust the importance of each weight accordingly. The substitution of random failed weights acts as a form of aggressive regularization, improving the network's generalization. As can be expected, training a neural network with stochastic jumps in weight values leads to a longer training time compared to a regular neural network, as is the case for Bayesian neural networks. However, this constant sampling also enables the network to be more robust to overfitting [79]. A pseudocode representing the forward pass process is also depicted in algorithm 1.

Comparison with regular neural network

Algorithm 1 HANN's forwardpass pseudocode

Input
W Weight matrix
B Bias matrix
X Input data
Output
Y Output data
$\epsilon_{cti} \leftarrow N\left(g_{s\pm}, f(g_{s\pm})^2\right) + N\left(\mu_{off}, \sigma_{off}^2\right)$
$\epsilon_{bse} \leftarrow g_{adj}(n_d)$
$W \leftarrow W + \epsilon_{cti} + \epsilon_{bse}$
if Substitution LRS ≥ 0 **then**
$mask_{LRS} \leftarrow$ randomly sampled indexes
end if
if Substitution HRS ≥ 0 **then**
$mask_{HRS} \leftarrow$ randomly sampled indexes
end if
$W[mask_{LRS}] \leftarrow N\left(\mu_{LRS}, \sigma_{LRS}^2\right)$
$W[mask_{HRS}] \leftarrow N\left(\mu_{HRS}, \sigma_{HRS}^2\right)$
$Y \leftarrow WX + B$

A simple half-moon problem was considered to compare the performance of our hardware-aware training approach with that of a regular neural network trained without taking into account hardware non-idealities. The two networks have the same learning rate α of 0.01, and they use an Adam optimizer [80] with a momentum of 0.9. The batch size used was 256. A neural network with one hidden layer with eight neurons is considered, and the sigmoid activation function is used (as depicted in Figure 3.1b). The cost function used is the binary cross-entropy loss [81]. During training, hyperparameter values of 0.5% were chosen for both the LRS-stuck and HRS-stuck substitutions. For the first layer, a recorded average of 3.1% of neural network weights were affected by random substitutions of LRS and HRS stuck devices during training. A half-moon classification problem training set of 875 data points and a test set of 200 data points were randomly generated. The neural network (NN) is trained without any consideration of the variability and then evaluated with the stochastic effects added to compare its performance to that of the hardware-aware NN (HANN). This is repeated n times to determine what percentage of the test set could be classified well over a given percentage of the n simulated transferred networks. These results are expressed in Table A.1 in Supplementary Note 4. A comparison of the accuracies is also displayed in Figure 3.5. Interpreting these results, we see that 79.5% of the test set can be properly classified by at least 95% of the simulated transferred hardware-aware networks. On the other hand, only 18.5% of the test set can be properly classified by at least

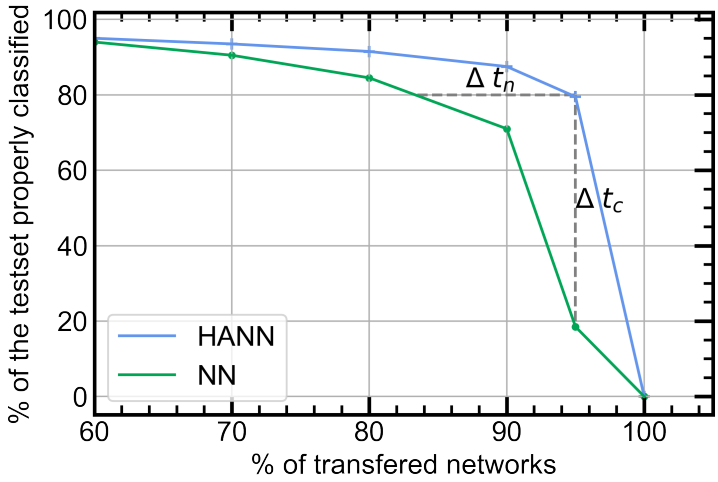

FIGURE 3.5 Comparison of the minimal accuracy over 10,000 simulated transfers achievable for the test set. None of the datapoints are successfully classified by 100% of the simulated neural networks due to the often aggressive substitution of weights by failing devices, large crosstalk effects and imprecise tuning. We can see that more of the test set is classified with a higher accuracy for the HANN (depicted in blue) compared to the NN (in green). For instance, 80% of the test set is properly classified by 83.5% of the simulated transferred regular networks vs 95% of the simulated transferred hardware-aware networks. This corresponds to an increase of $\Delta t_n = 11.5\%$. Reading the graph in the opposite way, we see that 95% or more of the simulated regular networks are capable of properly classifying only 18.5% of the test set, while 95% of the hardware-aware networks are capable of properly classifying 79.5% of the test set. This corresponds to an increase of $\Delta t_c = 61\%$.

95% of the simulated transferred regular neural networks. We observe that 87.5% of the test set can be properly classified by at least 90% of the simulated transferred hardware-aware networks, while 71% of the test set can be properly classified by at least 90% of the regular networks. The hardware-aware neural network training techniques contribute significantly to maintaining a high consistency and accuracy between simulated transferred networks. Figure 3.6 offers a graphical interpretation of the difference in the relative variability by showing the results of the test set classifications over 1,000 simulated transfers for each of the two networks, along with a heatmap representation of the variability with respect to the entire data space for the half moons dataset. Note that this is for a single instance of a hardware-aware-trained NN and a regularly trained NN. The variability, depicted by a deeper blue, is greater for the regular neural network and, as expected, is also greater at the frontier between the two classes.

3.2.4 Conclusion

It was demonstrated in this paper that the use of hardware-aware training techniques can make neural networks significantly more robust to their often difficult transfer onto passive

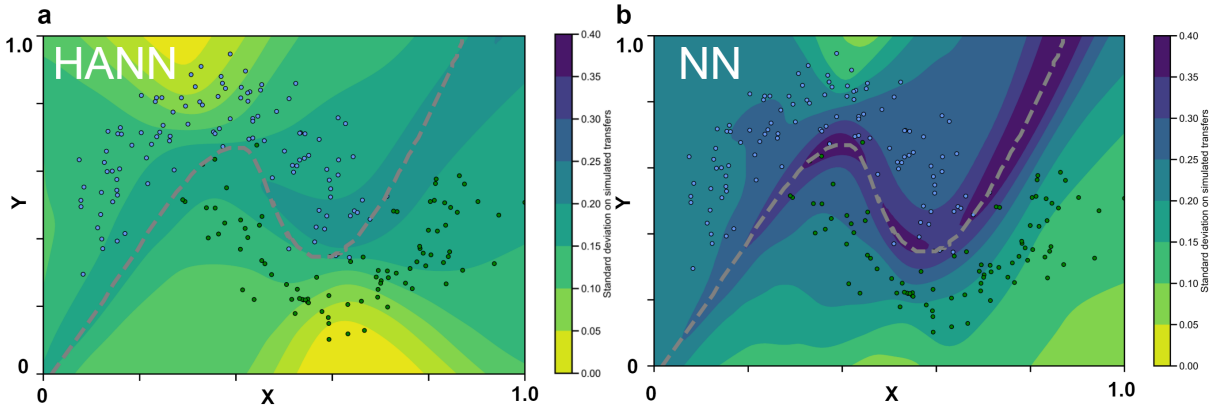

FIGURE 3.6 Heatmap of the variability of classifications with respect to the coordinates (x, y) used as input across the data space. a. Variability of the hardware-aware neural network. **b.** Variability of the regular neural network. The inference is repeated 1,000 times and the mean and standard deviation of the classified results (always 0 or 1) are obtained. A mean smaller than or equal to 0.5 corresponds to the blue class and a mean greater than 0.5 corresponds to the green class. The standard deviation is depicted in the heatmap, with a deeper blue corresponding to more variability across classifications. Note that both subfigures share the same color scale to make it possible to visually compare them. Most data points for the regular neural network lie within regions of greater variability than for the HANN. A dashed gray line that separates the two classes with a precision of 98% is provided for reference.

crossbar arrays. This protocol integrates experimentally measured non-idealities such as the conductance tuning imprecision, biasing scheme effects and device failures into the neural network training.

A classification task using the half moons dataset reveals that our hardware-aware training protocol results in a 95% accuracy across simulated transfers for 79.5% of the test set, compared to only 18.5% for the regular neural network. The hardware-aware training procedure could enable the use of passive crossbars despite their non-idealities for inference tasks and proposes a parallel solution to the usual fabrication process improvements. It may serve as a middle ground between fabrication and software solutions to accelerate practical, real-life applications of memristive devices. Furthermore, setting up a custom hardware-aware neural network architecture using preexisting libraries proves to be relatively straightforward and inexpensive in terms of hardware.

While the devices studied here are TiO_2 passive crossbars, techniques like this are applicable to different memristive technologies as long as a new database is built using the same procedures presented in this article. A future study should focus on evaluating the

scalability of this approach, with respect to the advantages (robustness, accuracy) and disadvantages (training time, variability-induced local minima). It would also be interesting to integrate the weight optimization strategy illustrated in reference [82] on top of the hardware-aware training to see if results could be further improved The potential of using this technique for more complex datasets should also be evaluated. Faster convergence during training could most likely be achieved by integrating the variability into the gradient descent procedure. This model could be improved by adding additional variability sources and by developing versatile libraries that make hardware non-idealities compatible with neural network requirements. The philosophy behind this work is to realistically render compatible in-memory computing for neural networks with the current non-negligible hardware non-idealities present in passive crossbar arrays. This demonstration, based on experimental data, is a step forward in making the use of passive crossbars for neural network matrix-vector multiplications viable in spite of their inherent shortcomings.

Methods

Tuning algorithm and characterization details

The pulse tuning algorithm presented in [76] was used. A memory was considered programmed when ten successive reading pulses returned a value within a tolerance around the target. The writing pulses were incremented from [0.8 V, 2 V] for SET pulses and from [-1.1 V, -2.4 V] for RESET pulses and had a duration of 200 ns. The read pulses had a duration of 10 μs and an amplitude of 0.2 V, while the current measurement range was 100 μA. The devices considered were forming free. Figure 3.2a depicts the conductance evolution using the pulse conductance tuning procedure. Devices that failed before the completion of the testing procedure were kept in the database, as this corresponds to a realistic scenario of a device's behavior before imminent failure, while devices that failed before testing began were kept in the HRS-stuck or LRS-stuck databases. More information regarding the characterization setup can be found in Supplementary Figure A.2 and Supplementary Note 2. A graph of hysteresis curves under DC stimulation for 129 of our devices is also available in Supplementary Figure A.3.

Fabrication of CMOS-compatible TiO_2-based memristor crossbar

The samples used for measurements were prepared as described in a previous publication [55]. First, a 600-nm-thick SiN layer was deposited on a Si substrate using low-pressure chemical vapor deposition (LPCVD). Electron-beam lithography (EBL) was used to pattern 400-nm-wide bottom electrodes, which were etched 150 nm deep using an inductively coupled plasma etching process with $CF_4/He/H_2$ chemistry. The trenches were filled

with 600-nm-thick TiN, deposited by reactive sputtering, and polished using chemical-mechanical polishing (CMP) to remove the excess TiN and planarize the samples. This resulted in embedded TiN electrodes within the SiN layer. The active switching layer was composed of 1.5 nm of Al_2O_3 and 15 nm of TiO_2, which were deposited using atomic layer deposition and reactive sputtering, respectively. EBL and plasma etching with $BCl_3/Cl_2/Ar$ chemistry were used to deposit and pattern 400-nm-wide top electrode lines with Ti (10 nm)/TiN (30 nm)/Al (200 nm). The switching layer outside the crossbar region was etched using the same process to suppress line-to-line leakages and open the ends of bottom electrodes. The crossbars were then encapsulated within a 500-nm-thick PECVD SiO_2 layer, and the contacts on the open ends of the top and bottom electrodes were etched using plasma etching with $CF_4/He/H_2$ chemistry. Ti (10 nm)/Al (400 nm) contact pads for wiring were deposited and patterned using EBL and plasma etching with $BCl_3/Cl_2/Ar$ chemistry. Finally, the sample was subjected to a rapid thermal annealing process at 350°C in N_2 gas for an unspecified duration.

Data availability

The hardware-aware training library, along with some of the experimental data, can be found in a GitHub repository that has been made publicly available at [83]. The rest of the data that support this work are available from the corresponding author upon reasonable request.

Acknowledgements

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds de Recherche du Québec Nature et Technologie (FRQNT). We acknowledge financial support from the EU : ERC-2017-COG project IONOS (# GA 773228). We would like to acknowledge Jonathan Vermette, Vincent-Philippe Rhéaume and the LCSM and LNN cleanroom staff for their assistance with the electrical characterization setup and nanofabrication process development. LN2 is a joint International Research Laboratory (IRL 3463) funded and co-operated in Canada by Université de Sherbrooke (UdeS) and in France by CNRS as well as ECL, INSA Lyon, and Université Grenoble Alpes (UGA). It is also supported by the Fonds de Recherche du Québec Nature et Technologie (FRQNT). We would also like to thank the IEMN cleanroom engineers for their help with the device fabrication.

Competing interests

The authors declare no competing interests.

CHAPITRE 4

CONCLUSION

4.1 Sommaire et contributions

Ce projet de recherche avait comme objectif principal d'instaurer une nouvelle méthode d'entraînement stochastique de réseau neuronal afin de mitiger les pertes de précisions accompagnant l'inférence sur crossbar passif de mémoires résistives. Cet entraînement fut réalisé à l'aide de techniques d'apprentissage classiques adaptées à la variabilité caractérisée des mémoires. L'objectif principale fut accompli tel que présenté dans la section résultats de l'article au cœur de ce mémoire. L'utilisation de la technique d'entraînement *hardware-aware* est manifestement plus efficace que l'entraînement classique afin d'assurer une meilleure robustesse au niveau des classifications ; avec comme principal résultat pour un problème de classification simple avec 79.5% des points de données bien classifiés pour 95% des transferts par rapport à seulement 18.5% des transferts pour le réseau classique. L'objectif principal fut accompli grâce à la réalisation des sous-objectifs : la quantification de la variabilité, son intégration subséquente à un modèle de réseau neuronal conscient de la variabilité et la comparaison de performances entre celui-ci et un réseau inconscient de la variabilité.

En termes des contributions originales, tel qu'il fût précédemment mentionné, c'est le premier réseau neuronal conscient de plusieurs sources de variabilité différentes (la précision de programmation des mémoires, l'effet du schéma de polarisation et les mémoires défaillantes) qui utilise l'architecture 0T1R, le schéma de polarisation V/3 et qui ne nécessite qu'une seule étape de programmation. La section 2.4 compare le projet à d'autres lui ressemblant et illustre ces contributions originales tandis que l'article du chapitre 3 relate leur succès.

L'une des principales faiblesses des réseaux neuronaux est leur comportement parfois imprévisible. Cet aspect de leur fonctionnement ralentit considérablement leur intégration dans des domaines nécessitant de la haute précision et de la robustesse considérable (tel que le domaine aérospatial ou le domaine médical). Dédoubler cette imprévisibilité d'une seconde variabilité matérielle rendrait leur intégration encore plus inadmissible pour ces domaines. Ceci, de pair avec leur consommation énergétique considérable, rendra l'utilisa-

tion de crossbar passif de mémoires résistives utilisé avec des réseaux neuronaux hardware-aware particulièrement séduisant pour des applications de systèmes embarqués et particulièrement pour ceux nécessitant une précision considérable. Ceci, premièrement pour limiter l'imprévisibilité de ces réseaux neuronaux analogues à un seuil tolérable et connu. Ensuite, pour limiter les pertes énergétiques de l'apprentissage machine habituel.

4.2 Perspectives

4.2.1 Reprogrammation probabiliste des mémoires

Il est fréquent [20] d'entreprendre un deuxième cycle de programmation des mémoires afin de rectifier les changements imprévus d'état de conductance lors de leur première programmation. Ces changements sont dus aux effets stochastiques énumérés dans l'article précédant cette section et dans la section de l'état de l'art. La méthode habituelle pour déterminer quelles mémoires seraient concernées par une étape de reprogrammation est de lire l'état conductif du crossbar entier et de comparer la conductance des dispositifs à leur conductance cible selon un seuil arbitrairement défini. Les mémoires ne répondant pas à ce critère sont alors reprogrammées. Par exemple, si une mémoire est à plus de 1% de différence de sa valeur numérique cible, elle est reprogrammée.

Cette approche, si utilisée directement et ainsi, serait contre-productive pour les poids du réseau stochastique HANN. Le réseau apprend à s'attendre à plus de variabilité par rapport à certains poids en fonction de leur état conductif et de leur position dans le crossbar. Le seuil précédemment mentionné devrait donc être établi en fonction de ces deux facteurs également. La question initialement posée lors du deuxième cycle de reprogrammation deviendrait : Est-ce que les poids synaptiques ont été programmés à une valeur de conductance tolérable et attendue ? En approximant les effets stochastiques ajoutés aux poids synaptiques par une distribution normale telle que représentée par l'équation 3.2 du chapitre 3, la règle 68-95-99.7 (ou loi empirique) pourrait être utilisée afin d'établir un seuil unique à chaque poids synaptique. Les poids lus après la programmation du crossbar devraient être bornés par $\mu \pm (3 \times \sigma)$ et donc se situer parmi 99.7% des valeurs attendues par la distribution de ce poids déterminée lors de l'entraînement. Cette approche servirait à contrer les imperfections du modèle stochastique intégré au réseau neuronal. Dans le monde réel, les mémoires ne se comporteront pas telles que simulées et il est fort possible qu'une mémoire présente une conductance après programmation ne correspondant à aucune conductance équivalente utilisée lors de son entraînement. En utilisant le deuxième cycle de programmation probabiliste, cette mémoire serait reprogrammée puisque son comportement aurait été anormal même en fonction de la variabilité prédite.

Une autre approche similaire serait l'utilisation d'un seuil hyperparamètre correspondant au z-score minimal de l'échantillon par rapport à sa valeur moyenne et de son écart-type pour décider quelles mémoires reprogrammer.

4.2.2 Intégration de la variabilité au gradient

Le réseau stochastique HANN s'apparente à un réseau bayésien du fait que ses poids sont puisés de distributions de la famille normale. Les réseaux neuronaux bayésiens sont entraînés en utilisant l'inférence variationnelle, ce qui fut négligé dans le cadre de l'article en cours de publication et étudié brièvement durant ce projet de maîtrise. L'algorithme d'entraînement de *bayes by backprop* est présenté dans la section B.0.1 et pourrait être utilisé pour faciliter la convergence du réseau vers une solution pour des problèmes complexes ou alors pour l'accélérer.

ANNEXE A

Informations supplémentaires

Supplementary Note 1 : Mathematical expression of range conversions between weights and conductances
The following equations represent the conversion of the weights to the conductance range, the addition of the noise from the reparametrization trick and the conversion back to the initial weight range. In these equations, $\phi_{\min} = \min(\phi)$ is the minimal weight value, $\phi_{\max} = \max(\phi)$ is the maximal weight value and $\phi_{\text{absmax}} = \max(|\phi|)$ is the maximal absolute value of the weights. $g_{max} = 400$ and $g_{min} = 100$, as described previously in this article. As shown in equation A.1, the weights are separated into negative and positive components, with all weight components of the opposite sign being set to 0 and negative weight values being converted into positive values. The weights are then converted through min-max scaling to the conductance range, as shown in equation A.2. An epsilon parameter that depends on the obtained positive and negative component conductance values is added to the weights, which are then converted back into the initial range, as shown in equation A.4. The value of the final ϵ parameter for the reparametrization trick, as shown in Figure 3.1b, is therefore represented by equation A.5 :

$$\begin{aligned} \phi &\mapsto \phi_+ - \phi_-, \\ \phi_+ &= \phi \geq 0, \\ \phi_- &= -\phi \geq 0, \end{aligned} \tag{A.1}$$

$$\phi_\pm \mapsto \frac{\phi_\pm - 0}{\phi_{\text{absmax}} - 0} \times (g_{\max} - g_{\min}) + g_{\min} = \phi_{g\pm}, \tag{A.2}$$

$$\phi_{g\pm} = \phi_{g\pm} + \varepsilon_{g\pm}\left(\phi_{g\pm}\right), \tag{A.3}$$

$$\begin{aligned} \phi_{g+} - \phi_{g-} &= \Delta\phi_g, \\ \Delta\phi_g &\mapsto \frac{(\phi_{g+} - \phi_{g-}) - (g_{\min} - g_{\max})}{(g_{\max} - g_{\min}) - (g_{\min} - g_{\max})}, \end{aligned} \tag{A.4}$$

$$\begin{aligned} &\times (\phi_{\max} - \phi_{\min}) + \phi_{\min} = \phi', \\ &\epsilon(\phi) = \phi - \phi'. \end{aligned} \tag{A.5}$$

Supplementary Note 2 : Electrical characterizations The electrical characterizations performed for the conductance tuning experiments were carried out with a Keysight

B1500 parameter analyzer. The waveform generator/fast measurement unit (WGFMU) module was used, along with a custom C++ script based on a Keysight library [84]. The instruments were controlled through a GPIB-USB connection. The crossbar programming and characterization were performed using a Keithley S4200 system attached to a custom pulse divider PCB and connected to a Keithley 707B switch matrix. The signals were then multiplexed onto a wire-bonded chip resting on another printed circuit board (PCB) platform, as depicted in Supplementary Figure A.2. A custom C library was created to be used with the *KULT*, *KITE* and *KCON* software suite on the Keithley.

Supplementary Note 3 : Comparative table of the results of the neural networks

Tableau A.1 Relative accuracies of every data point in the test set for both the hardware-aware neural network (HANN) and the regular neural network (NN) over 10000 simulated transfers. These points correspond to the points in Figure 3.5.

Percentage of simulated networks that correctly classify the datapoints(%)	**NN #data points**	**HANN #data points**
100	0	0
$95 \leq x < 100$	37 (18.5%)	159 (79.5%)
$90 \leq x < 95$	105 (52.5%)	16 (8%)
$80 \leq x < 90$	27 (13.5%)	8 (4%)
$70 \leq x < 80$	12 (6%)	4 (2%)
$60 \leq x < 70$	7 (3.5%)	3 (1.5%)
$50 \leq x < 60$	6 (3%)	2 (1%)
$x \leq 50$	6 (3%)	8 (4%)
Total	200 (100%)	200 (100%)

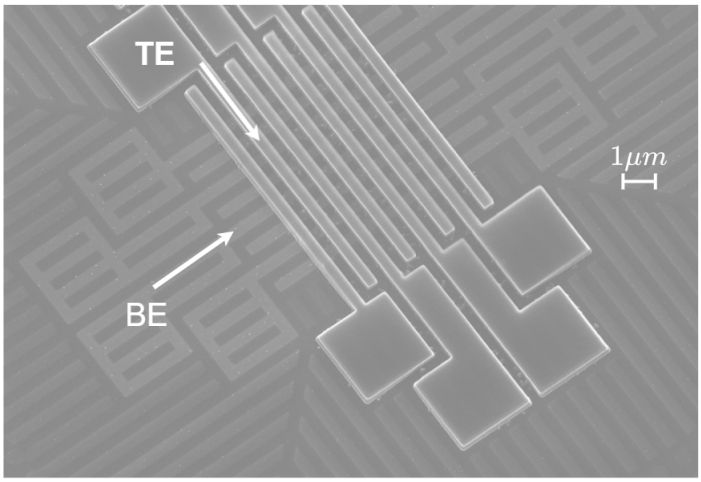

Figure A.1 Picture of a crossbar after top electrode patterning. Close-up view of the crossbar structure of our TiO_2-based devices after top electrode fabrication. The top electrode (TE) and bottom electrode (BE) are indicated on the picture. The pattern on the bottom electrode is implemented in order to improve the chemical mechanical polishing (CMP) employed during the damascene process.

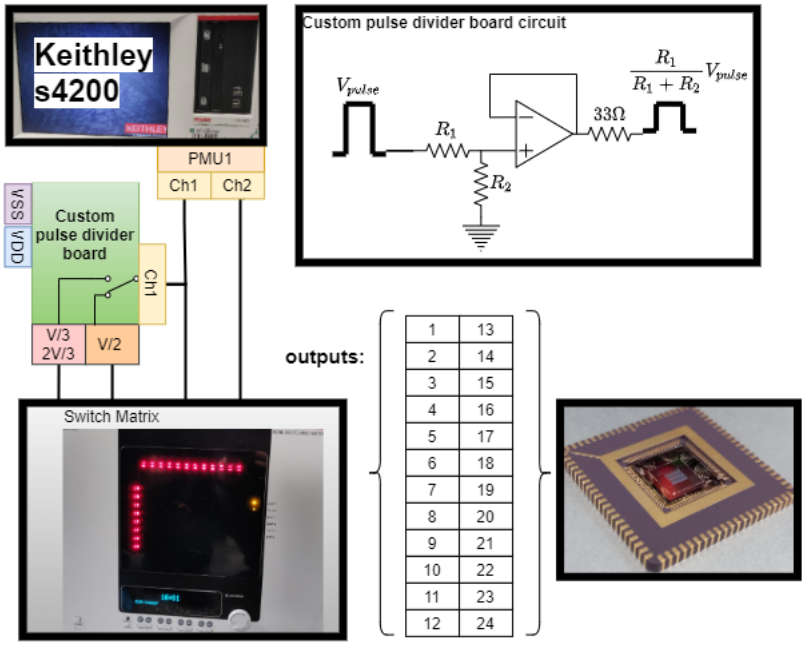

Figure A.2 Electrical characterization setup for crossbar measurements. A simple current follower circuit is used to divide the pulse into fractions of itself (which can be V/2 or V/3 and 2V/3). The outputs of the switch matrix are then connected to a wire-bonded chip through a series of coaxial cables.

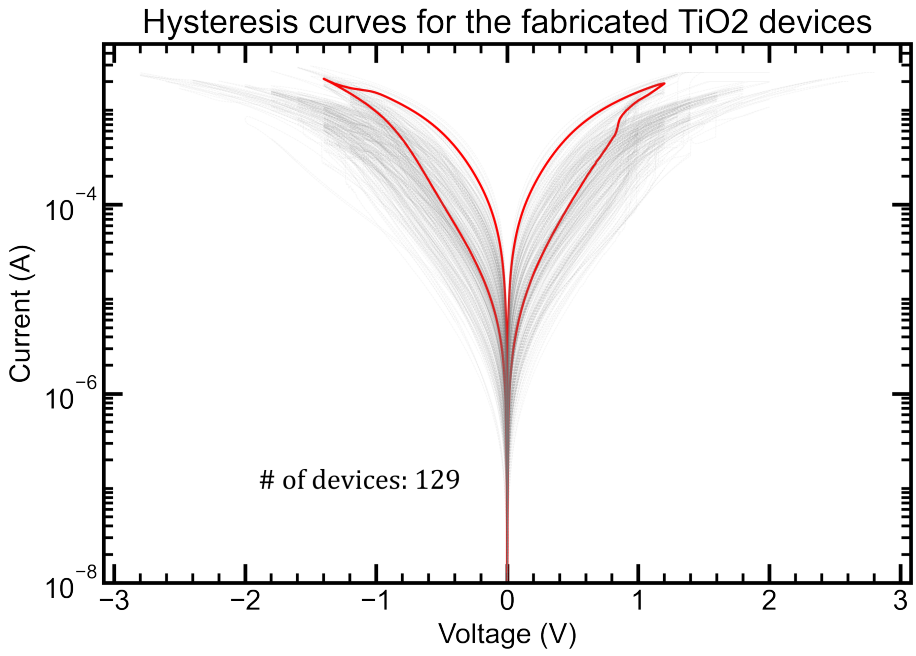

FIGURE A.3 Representation of the DC cycling of 129 of our devices (in crosspoint configuration). Devices were cycled using DC voltages reaching up to -3 V for RESET and 3 V for SET. One device's cycling is highlighted in red for reference. Only working devices are considered.

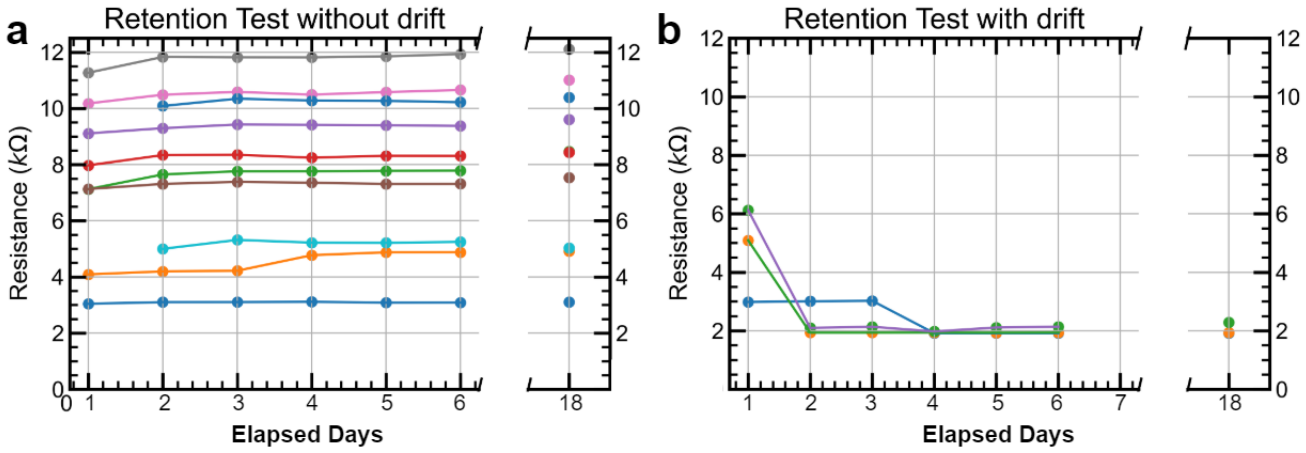

FIGURE A.4 Endurance test drift leading to LRS failure. a. Conductance evolution through time of devices that did not drift for the duration of the test **b.** Conductance evolution through time of devices that drifted during the test. Devices were programmed to different values within their resistive ranges and their resistances were read once per day for 18 days ; three of the devices showed drift towards the LRS. A read voltage of 0.2 V was used. This demonstrates that retention failures are LRS-type failures.

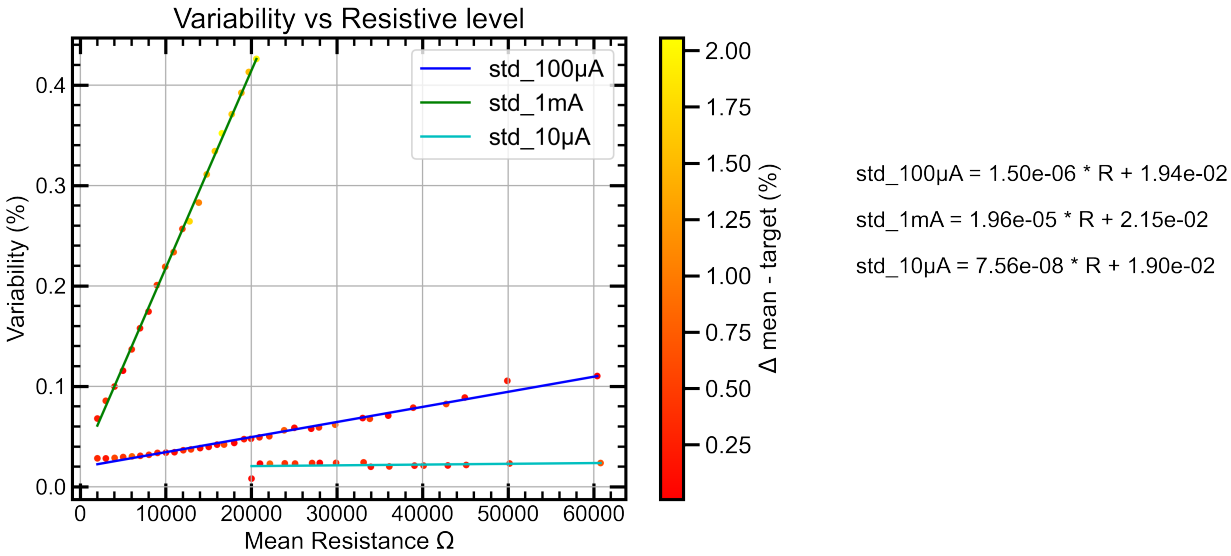

Figure A.5 B1500 read variability. Several through-hole resistances were read 1e6 times between 1k Ω and 60k Ω to measure the inherent variability of the instrument with respect to the resistive value. The y value represents the standard deviation of the read values (as a percentage) and the heatmap on the right represents the drift of the mean of the measured values from the actual value of the resistor (note that the manufacturer guaranteed a maximal offset of 0.1% of the manufactured resistances from their target values). The variability is expressed as a percentage of the mean. The three differently colored lines represent the measurement ranges of the instrument. For the studied range of conductances, the variability never reaches more than 0.05% of the mean value for the 100 μA current range used.

ANNEXE B

Littérature Bayésienne

B.0.1 Réseaux neuronaux bayésiens

Les réseaux bayésiens sont une forme de réseaux neuronaux utilisant des distributions probabilistes en guise de paramètres. Leur principe de fonctionnement est de prélever des poids synaptiques d'une distribution probabiliste au lieu de leur assigner des valeurs discrètes (figure 2.1 c). Ce principe les rend significativement plus résistants au surapprentissage (*overfitting*) que les réseaux neuronaux habituels (sans toutefois les en immuniser [85]). Certaines formes de réseaux neuronaux bayésiens utilisent des fonctions d'activation probabilistes au lieu de poids (tel que montré dans la figure 2.1 b). Cette option peut permettre la réduction du nombre de paramètres d'optimisation au coût d'être significativement plus complexe à entraîner.

Il est important de garder en tête que la précision des réseaux neuronaux bayésiens n'est pas supérieure à celle d'un réseau neuronal régulier. Leur pertinence (ainsi que celle des réseaux neuronaux stochastiques en général) provient de leur capacité à caractériser l'incertitude de leurs inférences. Leur robustesse les rend particulièrement utiles dans le domaine médical [86, 87, 88] et les rend capables de distinguer des images régulières d'images contradictoires (voir figure B.1). Les domaines de l'ingénierie nécessitant l'édification d'un processus décisionnel éthique ont intérêt à s'armer d'outils mathématiques bayésiens capables de quantifier leur incertitude (le domaine des voitures autonomes en est un bon exemple [89]).

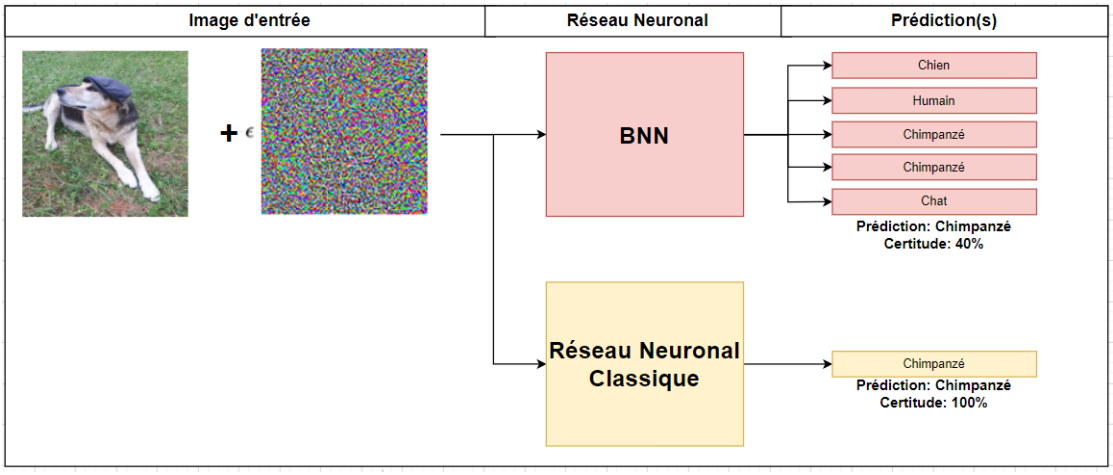

Figure B.1 Illustration des prédictions potentielles provenant d'un réseau neuronal bayésien contre celle d'un réseau neuronal classique vis-à-vis une image contradictoire. La prédiction finale est la même, mais l'incertitude l'entourant permet de prendre une décision plus éclairée face à celle-ci.

Inférence bayésienne

La loi de Bayes figurant dans l'équation B.1 est au coeur des réseaux neuronaux bayésiens :

$$P(H \mid D) = \frac{P(D \mid H)P(H)}{P(D)} = \frac{P(D \mid H)P(H)}{\int_H P\left(D \mid H'\right) P\left(H'\right) dH'} = \frac{P(D,H)}{\int_H P\left(D,H'\right) dH'} \tag{B.1}$$

Où, $P(D|H)$ (en anglais : *the likelihood*) est la probabilité a posteriori de D sachant H , $P(H)$ est la probabilité à priori de H, $P(D)$ est la probabilité à priori de D (la preuve ou encore, en anglais : *the evidence*) et $P(H|D)$ est la probabilité a posteriori de H sachant D. Si un système d'évènements H est considéré et que tous les évènements composant ce système sont mutuellement exclusifs et lorsque réunis, exhaustifs, la preuve peut se réécrire : $\int_H P\left(D,H'\right) dH'$. L'interprétation classique de cette loi veut que H soit une hypothèse et que D représente les données. Pour un réseau neuronal bayésien, l'hypothèse (H) devient les paramètres du réseau, représentés par w. Ainsi, le théorème de Bayes peut se réécrire :

$$p(w \mid D) = \frac{p\left(D_y \mid D_x, w\right) p(w)}{\int_w p\left(D_y \mid D_x, w'\right) p\left(w'\right) dw'} \propto p\left(D_y \mid D_x, w\right) p(w) \tag{B.2}$$

Où Dy représente l'étiquette des données alors que Dx représente leur valeur numérique. Il est difficile de déterminer une distribution à priori informée sur les poids synaptiques d'un réseau neuronal bayésien. Pour cette raison, il faut contrer l'influence de la distribution à priori mal ou non-informé sur les paramètres en augmentant considérablement la quantité de données du jeu de donnée lors de l'entraînement (ce qui équivaut à favoriser le *likelihood*). Pour les problèmes à plusieurs dimensions, il est évident qu'à cause de la preuve, échantillonner depuis cette distribution ainsi que la calculer directement est trop demandant numériquement. Deux techniques sont donc habituellement utilisées pour l'approximer : Monte-Carlo par chaîne de Markov et l'inférence variationnelle. Après avoir trouvé $p(w \mid D)$ en utilisant l'une de celles-ci, il devient possible de déterminer la distribution de probabilité marginale des données de sortie :

$$p(\boldsymbol{y} \mid \boldsymbol{x}, D) = \int_{\boldsymbol{w}} p\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{w}'\right) p\left(\boldsymbol{w}' \mid D\right) d\boldsymbol{w}' \tag{B.3}$$

Quelques échantillons de w (dénotés θ) suffisent généralement à son estimation. Lors d'un problème de régression, la moyenne du modèle (NN_θ) sur N échantillons est considérée comme solution :

$$\hat{\boldsymbol{y}} = \frac{1}{|N|} \sum_{\theta_i \in N} NN_{\theta_i}(\boldsymbol{x}) \tag{B.4}$$

Lors d'un problème de classification, la valeur de $\hat{\boldsymbol{y}}$ représente la probabilité de chaque classe et la probabilité moyenne maximale est la prédiction finale. Une comparaison des résultats individuels découlant des N échantillons permet de comprendre l'incertitude associée aux données de sortie.

Méthode de Monte-Carlo par chaîne de Markov

Anciennes sans être archaïques, les méthodes de MCMC persistent à s'adapter aux exigences informatiques titanesques des jeux de données modernes [90]. À la base, elles construisent une chaîne de Markov d'échantillons aléatoires dépendant de celui les précédant directement afin d'obtenir une approximation adéquate de la distribution à posteriori de l'équation B.2. La distribution depuis laquelle les échantillons sont tirés doit être proportionnelle et toujours supérieur à la distribution a posteriori véritable. Généralement, la fonction utilisée est une multiplication de la probabilité à posteriori de D selon W à la preuve tel que représenté au côté droit de l'équation B.2. Avant d'obtenir des échantillons approchant bien la distribution a posteriori, il y a une période difficilement déterminable appelée le *burn-in* d'échantillons sacrificiels n'appartenant pas à la distribution approximative désirée. En effet, puisque les premiers échantillons sont tirés au hasard, cela peut prendre une quantité considérable d'échantillons avant d'en obtenir appartenant à la distribution. Il est également tout-à-fait possible que les premiers échantillons appartiennent déjà à la distribution recherchée. Le cas échéant, le rejet de ces échantillons peut être dommageable à la qualité de l'approximation. Il y existe donc une certaine ambiguïté dans la littérature par rapport à l'utilité de rejeter les échantillons de *burn-in* [91, 92]. Cette technique requiert également beaucoup d'échantillons Θ à préserver en mémoire avant d'estimer adéquatement la distribution à posteriori des poids, car le θ de l'équation B.3 est échantillonné uniformément dans cet ensemble Θ lors de l'inférence. La classe de méthode MCMC la plus appropriée pour les réseaux neuronaux bayésiens est celle de Metropolis-Hastings [93] de laquelle découlent d'autres techniques plus complexes telles que NUTS-MCMC [94] (No U-Turns Markov Chain Monte-Carlo) et Monte-Carlo Hamiltonien. Son principe de fonctionnement est de toujours accepter un paramètre échantillonné si sa probabilité à posteriori est plus grande que celle du paramètre le précédant et de l'accepter avec une probabilité p sinon.

Les principales faiblesses de MCMC sont la dépendance de la distribution a posteriori estimée aux premières valeurs de la chaîne de Markov (le *burn-in*), le phénomène d'autocorrélation y survenant et la quantité d'échantillons à garder en mémoire. Une technique de réduction de l'autocorrélation est l'amincissement (*thinning*) ; ce qui consiste simplement à n'accepter qu'un seul échantillon sur N [95]. Cela augmente malheureusement de N fois le temps nécessaire pour déterminer la distribution a posteriori des paramètres.

Inférence variationnelle

Pour un réseau neuronal bayésien, l'inférence variationnelle optimise des paramètres (θ) qui minimisent la distance de Kullback-Leibler [96] entre la distribution des poids selon eux $q(w \mid \theta)$ et leur véritable distribution a posteriori $p(w \mid D)$ (première ligne de l'équation B.5). Au premier abord, il semblerait y avoir ici un problème de «la poule et de l'oeuf » ; la véritable distribution $p(w \mid D)$ est nécessaire à sa propre estimation. Cependant, remplacer $p(w \mid D)$ dans l'équation de KL par son équivalent bayésien ($\frac{p(D|w)p(w)}{p(D)}$) et appliquer quelques simples manipulations algébriques découlant du fait que p(D) est constant par rapport à w et que l'intégrale de $q(w|\theta)$ par rapport à w est égale à 1 (puisque c'est une distribution) la transforme en un équivalent indépendant de la véritable distribution a posteriori (voir équation B.5). L'inverse des deux premiers termes à sa dernière ligne est

connu sous le nom de *evidence lower-bound* ou encore *expected lower-bound* (l'ELBO). La divergence de Kullback-Leibler sera toujours plus grande ou égale à 0 [97].

$$\begin{aligned}
\mathrm{KL}(q(w \mid \theta)) \| p(w \mid D) &= \int q(w \mid \theta) \log \frac{q(w \mid \theta)}{p(w \mid D)} \mathrm{d}w \\
&= \int q(w \mid \theta) \log \frac{q(w \mid \theta) p(D)}{p(D \mid w) p(w)} \mathrm{d}w \\
&= \int q(w \mid \theta) \log \frac{q(w \mid \theta)}{p(D \mid w) p(w)} \mathrm{d}w + \int q(w \mid \theta) \log p(D) \mathrm{d}w \\
&= \int q(w \mid \theta) \log \frac{q(w \mid \theta)}{p(w)} \mathrm{d}w - \int q(w \mid \theta) \log p(D \mid w) \mathrm{d}w + \log p(D) \\
&= \mathrm{KL}(q(w \mid \theta) \| p(w)) - E_q[\log p(D \mid w)] + \log p(D)
\end{aligned} \tag{B.5}$$

Puisque $\log p(D)$ est indépendant de w, il est possible de l'ignorer lors de la maximisation. *Mean-field variational bayes* (MFVB), une technique fréquemment utilisée pour trouver une distribution $q(w \mid \theta)$ pertinente à l'approximation par inférence variationnelle, stipule que la distribution a posteriori des poids est une composition de sous-distributions indépendantes et facilement caractérisables. L'hypothèse est présentée mathématiquement dans l'équation B.6, dans laquelle J représente le nombre de poids synaptiques du réseau et q_j est une distribution appartenant à la famille exponentielle (gaussienne, beta, log-normal, etc.).

$$Q_{MFVB} := \left\{ q : q(w \mid \theta) = \prod_{j=1}^{J} q_j \left(w_j \mid \theta_j \right) \right\} \tag{B.6}$$

Cette approximation permet l'optimisation de la distribution (selon le coût de Kullback-Leibler) en la rendant dépendante de valeurs discrètes de μ, σ, etc. facilement modifiables (lors d'une rétropropagation de gradients par example). Deux exemples de techniques d'optimisation de la distribution approximative $q(w \mid \theta)$ selon MFVB sont *Stochastic variational inference* [98] et *Automatic Differentiation Variational Inference*. [99].

Bayes by Backprop

La méthode de Bayes by Backprop [100] est une dérivée pratique de MFVB cherchant à accorder les réseaux neuronaux bayésiens avec la descente de gradient (*Gradient descent*) de l'apprentissage machine. Chaque poids synaptique (ou biais), possède deux paramètres : μ et ρ. Le paramètre μ représente sa valeur moyenne tandis que ρ est relié à son écart-type (σ) ainsi : $\sigma = \log(1 + \exp(\rho))$. Cela assure que l'écart-type σ ne soit jamais négatif (ce qui serait susceptible d'arriver suite à de trop nombreuses soustractions successives lors de la rétropropagation des gradients si σ était directement un paramètre). Ainsi, l'hypothèse de MFVB de BbyB est que les sous-distributions de la distribution a posteriori des paramètres sont normales.

BbyB propose d'exploiter une variable $\epsilon \sim q(\epsilon)$ en tant que source de bruit stable. Les paramètres variationnels θ de la distribution (autrement dit, les échantillons de w) sont ensuite échantillonnés selon ϵ à l'aide d'une transformation déterministe $t(\theta, \epsilon) = \mu + \log(1 + \exp(\rho)) \circ \epsilon$. En estimant que $q(w|\theta)$ répond au critère suivant : $q(\epsilon)d\epsilon = q(w|\theta)dw$, il est possible de déclarer que pour une fonction f différentiable selon w :

$$\frac{\partial}{\partial\theta} E_{q(\mathbf{w}|\theta)}[f(\mathbf{w}, \theta)] = E_{q(\epsilon)}\left[\frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \theta} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \theta}\right] \tag{B.7}$$

La preuve de cette proposition figure dans l'article. Elle est en fait une généralisation de l'astuce de reparamétrage gaussien [101] (*gaussian reparametrization trick*). En choisissant un coût approximatif :

$$\mathcal{F}(\mathcal{D}, \theta) \approx \sum_{i=1}^{n} \log q\left(\mathbf{w}^{(i)} \mid \theta\right) - \log(P\left(\mathbf{w}^{(i)}\right) P(\mathcal{D} \mid \mathbf{w}^{(i)})) \tag{B.8}$$

et en choisissant la fonction f de l'équation B.7 comme étant l'intérieur de la sommation de l'équation B.8, il devient possible de calculer les gradients :

$$\begin{aligned} \Delta_\rho &= \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho)} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \rho} \\ \Delta_\mu &= \frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \theta)}{\partial \mu} \end{aligned} \tag{B.9}$$

Or, selon cette définition de $f(\mathbf{w}, \theta)$, l'expression : $\frac{\partial f(\mathbf{w}, \theta)}{\partial \mathbf{w}}$ correspond exactement au gradient de l'équation 2.2. Cela signifie que le gradient de BbyB n'est en fait que le gradient calculé lors de la rétropropagation des gradients d'un réseau neuronal classique multiplié et décalé. La distribution à priori P(w) utilisée par BbyB est un mélange de deux gaussiennes (*scale mixture model*) centrées à zéro avec deux écarts-types différents :

$$P(\mathbf{w}) = \prod_j \pi \mathcal{N}\left(\mathbf{w}_j \mid 0, \sigma_1^2\right) + (1 - \pi)\mathcal{N}\left(\mathbf{w}_j \mid 0, \sigma_2^2\right) \tag{B.10}$$

Dans l'équation B.10, w_j représente le poids synaptique j du réseau et π est un paramètre permettant au besoin de favoriser l'une des deux distributions composant la distribution à priori ($0 \leq \pi \leq 1$). Cette décision donne à la distribution à priori une apparence de *spike-and-slab* dont l'aspect est modifiable selon les valeurs d'écart-type choisies et la corrèle aux poids synaptiques de manière à ce qu'elle n'ait pas à être optimisée par rapport aux données.

ANNEXE C
Technique Ping-Pong

C.0.1 Intérêt

Les réseaux neuronaux fondés sur l'inférence bayésienne se démarquent entre autres pour les applications médicales et pour les systèmes de voitures autonomes, car leur processus décisionnel prend en compte l'incertitude (voir section B.0.1) . Cette prise en compte de l'incertain entraîne cependant une augmentation du nombre de paramètres et nécessite une coûteuse génération de nombres aléatoires. Une approche prometteuse pour détourner ce problème serait d'utiliser les mémoires résistives. Habituellement, les défauts inhérents dont celles-ci sont affligées les rendent difficiles à utiliser en tant que poids synaptique discret (d'où l'intérêt du réseau conçu dans le cadre de ce projet). Cependant, des approches intéressantes ont récemment été proposées afin d'utiliser leurs aspects aléatoires pour pratiquer de l'inférence bayésienne. De telles approches ont le potentiel de réduire leur consommation énergique tout en augmentant la vitesse de l'inférence bayésienne. L'approche *Ping-Pong* proposée suggère de programmer des mémoires à la valeur μ de leur poids synaptique respectif et puis d'échantillonner autour de ce μ en appliquant deux pulses de polarité opposée modifiant l'état conductif de la mémoire selon un σ approximatif caractérisé (voir figure C.1). Le succès de cette technique permettrait de s'affranchir de la corrélation existante entre variabilité et niveau résistif des mémoires et permettrait un échantillonnage d'une matrice de poids synaptiques complète en parallèle en n x (3 x 200ns) où n est le nombre de rangées (WL) du crossbar et où un temps de pulse et un temps de repos entre chaque pulse de 200ns est assumé. La viabilité de l'approche en question nécessiterait une uniformité d'échantillonnage entre les différentes mémoires et une répétabilité des distributions obtenues adéquates. Par exemple, il devrait être possible de déterminer les paramètres de la distribution normale de laquelle l'application de deux pulses de 1.5V et de -1.5V viendrait échantillonner (μ_{1V5} et σ_{1V5}). Cette distribution devrait rester uniforme dans le temps et entre les mémoires résistives d'un même crossbar afin d'être mesurable préinférence. Un exemple d'une telle distribution est présenté dans la figureC.1 b tandis qu'un exemple d'échec de la technique à cause d'un changement de conductance prononcé est présenté dans la figureC.1 c. Il serait alors possible de créer une base de données répertoriant l'équivalence entre amplitude des pulses positifs et négatifs (ou même en fonction de la durée) et leur σ résultant.

C.0.2 Résultats

Dans les faits, la variabilité au travers le temps et entre les mémoires rend l'utilisation de la technique irréaliste dans l'immédiat. Pour pouvoir utiliser la technique ping-pong efficacement, il faudrait que les distributions normales $N(\mu, \sigma)$ atteignables soient :

- Toujours modélisable par une distribution normale.
- Modulables au niveau du σ
- Reproductibles entre les niveaux de conductances représentant le μ
- Reproductibles entre les différentes mémoires résistives d'un même crossbar

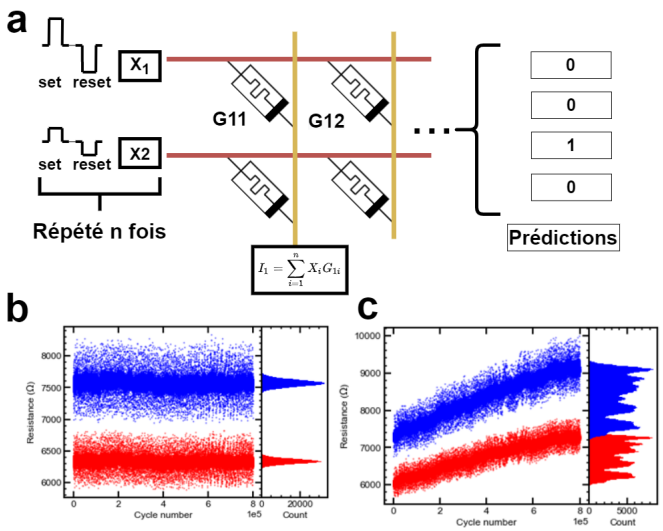

Figure C.1 **a**. Exemple simplifié de la technique *ping-pong* appliquée sur crossbar et de l'effet potentiel de celle-ci sur les prédictions d'un réseau traitant un jeu de données de deux classes (0 et 1). **b.** Exemple de la distribution résistive des mémoires après l'application de deux pulses de polarités opposées de 1.1V avec aucun drift après 8e5 cycles. **c.** Exemple de la distribution résistive des mémoires après l'application de deux pulses de polarités opposées de 1.1V avec un drift significatif après 8e5 cycles

Les distributions atteignables ne possèdent présentement aucune de ces propriétés tel que démontré dans la figure C.2. La sous-figure C.2 a représente le comportement idéal des mémoires pour permettre l'application de cette méthode. Les sous-figures C.2 b, C.2 c et C.2 d, elles, représentent le véritable comportement des mémoires lors de l'application de pulses opposées. Le manque d'uniformité dans les résultats obtenus a obstrué l'avancement de cette technique et a mené à l'abandon du projet pour l'instant.

C.0.3 Perspectives

Une étude subséquente de cette technique sera envisageable lorsque les dispositifs présenteront une plus grande uniformité entre eux et dans leur comportement au niveau de la programmation. Dès lors, il deviendra plus pertinent de refaire les tests illustrés. Il y aura sans doute une plus grande uniformité des comportements.

Entre temps, un entraînement de réseau neuronal avec des écart-types précis et corrélés à ceux atteignables présentement serait envisageable.

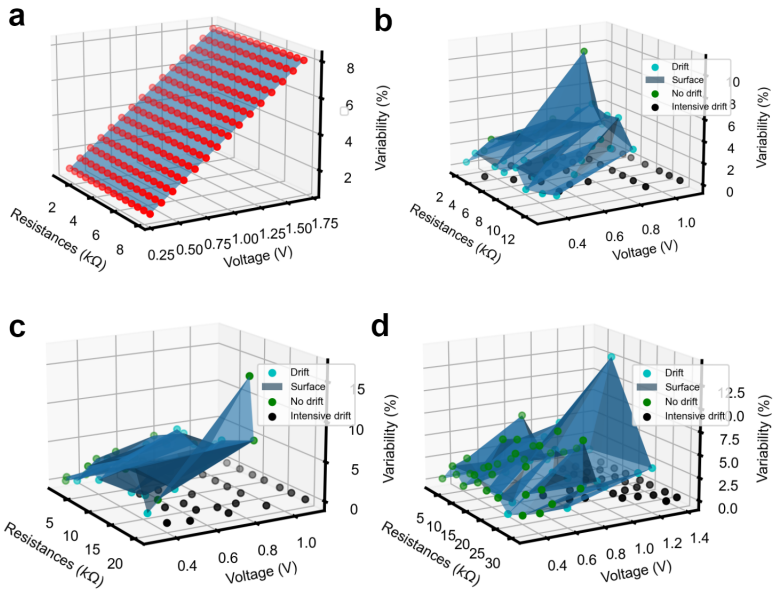

Figure C.2 **a)** Résultat idéal de la technique. La variabilité est uniforme en fonction de la résistance (la technique est donc applicable pour n'importe lequel μ) et une plage de variabilité adéquate pour représenter des poids synaptiques bayésiens tel qu'entraînés par *bayes by backprop* est atteignable. **b,c,d)** Résultats expérimentaux illustrant le véritable comportement de la technique. Il n'y a aucune uniformité entre les variabilités atteignables en fonction de la résistance pour une même mémoire et entre elles. L'amplitude des variabilités diffère similairement. La majorité des points testés divergent de la valeur μ attendue de plus que 5% de la valeur initiale (tel qu'illustré dans la figure C.1 c) et représenté dans les figures par un point noir. Les points bleue représentent un décalage de la valeur μ après 1e5 itérations. Toutes les mesures divergent après 1.1V d'amplitude

ANNEXE D

Modèle RNN de Memristor

D.0.1 Intérêt

Dans l'optique de simuler le comportement des mémoires afin d'élaborer des preuves de concept robustes, il fallut choisir un modèle de mémoire résistive approprié. Plusieurs modèles existants sont particulièrement intéressants et compatibles avec nos mémoires résistives à base d'oxyde de titane. Le modèle *data-driven*[102, 103] le modèle IM2NP [104], le modèle Stanford-PKU[105] et le récent modèle compact de dynamique de permutation sous l'effet de dépendance à la température [106, 107] en sont quelques exemples. Or, ces modèles n'arrivent souvent pas à généraliser le comportement des mémoires ou sont simplement trop demandant numériquement pour être applicable à des simulations d'ampleur considérable. La quantité de données amassées lors des caractérisations dans le cadre de cette maîtrise inspira un effort de conception d'un réseau neuronal récursif de petite taille capable de générer un comportement simulé réaliste pour une ou plusieurs mémoires résistives vis-à-vis un train de pulses de programmation. Le modèle pourrait dès lors apprendre les dynamiques asymétriques de commutations des mémoires dépendant de l'état conductif de celles-ci. Cette asymétrie est notamment visible sur la sous-figure D.1 a, le SET étant plus puissant que le RESET et la résistance tend vers son LRS après l'application d'un nombre équivalent de pulses positifs et négatifs.

Le réseau prenant en entrée un historique des n dernières caractéristiques (amplitude de la tension, durée, dernière état resistif, etc.) des pulses (où n est un hyperparamètre) détermine le changement résistif à t+1 et replace en entrée les sorties de cette inférence pour la simulation du prochain pulse en décalant les n-1 autres pulses. Le caractère récursif du réseau lui permet de garder en mémoire un état caché dénoté h considérant sommairement les pulses précédant les n derniers pulses précédemment mentionnés. Les grands avantages de ce réseau sont donc sa capacité à apprendre la relation entre l'état de conductance, l'historique des pulses et ses dynamiques de commutation (tel qu'il fût mentionné plus tôt) et sa capacité à traiter tout les mémoires du crossbar en *minibatch* et d'ainsi accélerer considérablement les simulations de grande échelle en s'appuyant sur les optimisations de calculs matricielles pré-existant et compatibles *GPU*. Or, sa plus grande faiblesse (encore non surmontée) est l'accumulation de son erreur provenant de la rétropropagation de son inférence à ses entrées. Cette erreur peut mener à des problèmes de divergence importants tel qu'il est montré dans la figure D.1 d.

D.0.2 Résultats initiaux

Les tests réalisés cherchaient à comparer les sorties du réseaux face aux données expérimentales et de déterminer l'écart entre ceux-ci. La résistance de l'axe des ordonnées est mise à l'échelle [0,1], afin d'aider le réseau à converger. Entre la sous-figure D.1 a et les sous-figures D.1 b, D.1 c et D.1 d, le réseau est également transformé en réseau stationnaire. C'est-à-dire que la caractéristique prise en entrée par le réseau relatant l'état résistif

de la mémoire devient le changement résistif depuis le dernier pulse au lieu de l'état résistif. Comme on peut le voir dans la figure D.1 b, les sorties du réseau ressemblent aux données expérimentales lorsqu'un seul état résistif initial est considéré. Dans les sous-figures D.1 a et D.1 b cependant, le réseau est entraîné à l'aide d'un jeu de données comportant le comportement d'une mémoire à l'application de plusieurs trains de pulses différent (un exemple de train de pulse serait de 0.2V à 1.8V en inversant la polarité à chaque 200 pulses et en incrémentant de 0.2V à chaque 400 pulses) et en débutant de plusieurs niveaux résistifs différents. Le réseau diverge plus des données expérimentales mais réussit à comprendre sommairement le comportement de la mémoire à différents états résistifs initiaux et en fonction de différents trains de pulses.

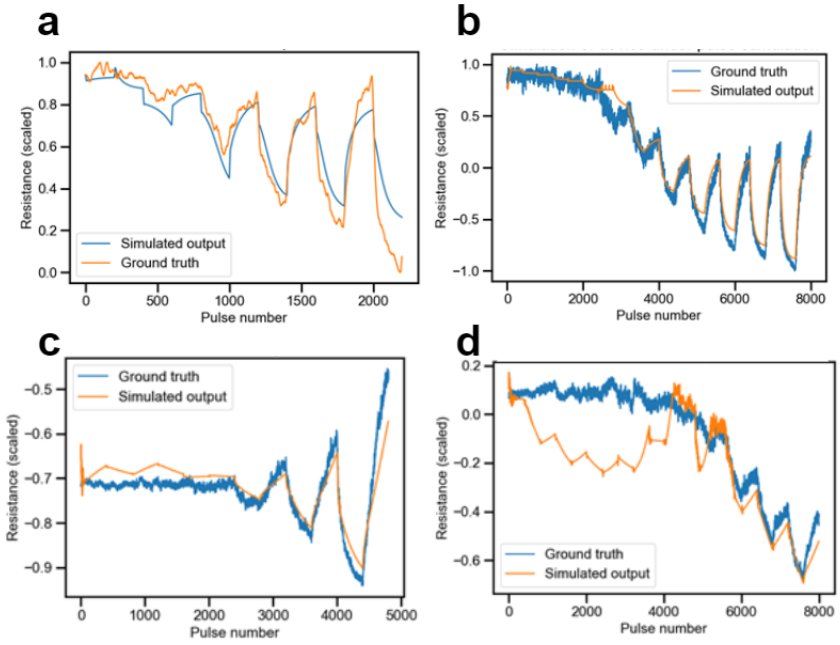

Figure D.1 **a)** Exemple de simulation du comportement d'une mémoire au travers 2500 pulses avec un jeu de données limité au comportement d'une mémoire à un seul niveau résistif initial. **b)** Exemple de simulation au travers 8000 pulses avec un jeu de donnée stationnaire limité également à un seul niveau résistif initial **c,d)** simulation d'un même réseau à deux niveaux résistifs initiaux et à deux trains de pulses différents. la sous figure c présente une assez bonne équivalence entre la simulation et les données expérimentales alors que la figure d diverge considérablement.

D.0.3 Perspectives

Malheureusement, afin de se concentrer sur l'apprentissage machine sur crossbar en pratique (création de l'architecture de test, programmation de l'instrument de mesure, etc.), le projet fût délaissé et devra être repris dans un futur rapproché. Les principales avenues pour faire avancer ce projet seront d'expérimenter avec des structures (nombre de couches et de neurones) de réseau plus importantes ainsi que d'étoffer le jeu de données pour l'entraînement. Le réseau devrait demeurer assez petit pour que son utilisation en tant que simulateur demeure pertinent pour des simulations à grande échelle.

LISTE DES RÉFÉRENCES

[1] Richard Bellman. *Dynamic programming*. Princeton Univ. Pr, Princeton, NJ, 1984.

[2] Big data needs a hardware revolution. *Nature*, 554(7691) :145–146, February 2018.

[3] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481) :984–986, February 2020.

[4] L. Chua. Memristor-the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5) :507–519, 1971.

[5] Dmitri B. Strukov, Gregory S. Snider, Duncan R. Stewart, and R. Stanley Williams. The missing memristor found. *Nature*, 453(7191) :80–83, May 2008.

[6] Hyungjin Kim, Hussein Nili, Mahmood Mahmoodi, and Dmitri B. Strukov. 4k-memristor analog-grade passive crossbar circuit. *CoRR*, abs/1906.12045, 2019.

[7] L. Goux, A. Fantini, G. Kar, Y.-Y. Chen, N. Jossart, R. Degraeve, S. Clima, B. Govoreanu, G. Lorenzo, G. Pourtois, D.J. Wouters, J.A. Kittl, L. Altimime, and M. Jurczak. Ultralow sub-500na operating current high-performance tin\al2o3\hfo2\hf \tin bipolar rram achieved through understanding-based stack-engineering. In *2012 Symposium on VLSI Technology (VLSIT)*, pages 159–160, 2012.

[8] Byung Joon Choi, Antonio C. Torrezan, John Paul Strachan, P. G. Kotula, A. J. Lohn, Matthew J. Marinella, Zhiyong Li, R. Stanley Williams, and J. Joshua Yang. High-speed and low-energy nitride memristors. *Advanced Functional Materials*, 26(29) :5290–5296, 2016.

[9] Alexander Vera-Tasama, Marisol Gomez-Cano, and J.I. Marin-Hurtado. Memristors : A perspective and impact on the electronics industry. In *2019 Latin American Electron Devices Conference (LAEDC)*, volume 1, pages 1–4, 2019.

[10] Intel. Neuromorphic computing - next generation of ai, 2021.

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch : An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[12] Prashant Gupta. Regularization in machine learning, Nov 2017.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[15] David E. Rumelhart, editor. *Parallel distributed processing. 1 : Foundations / David E. Rumelhart*. MIT Pr, Cambridge, Mass, 12. print edition, 1999.

[16] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. Generative adversarial networks : An overview. *IEEE Signal Processing Magazine*, 35(1) :53–65, 2018.

[17] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2) :29–48, 2022.

[18] Jiyong Woo, Jeonghwan Song, Kibong Moon, Sangheon Lee, Jaesung Park, and Hyunsang Hwang. Multilevel conductance switching of a hfo2 rram array induced by controlled filament for neuromorphic applications. In *2016 IEEE Silicon Nanoelectronics Workshop (SNW)*, pages 40–41, 2016.

[19] Heba Abunahla, Baker Mohammad, Maguy Abi Jaoude, and Mahmoud Al-Qutayri. Novel hafnium oxide memristor device : Switching behaviour and size effect. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017.

[20] Sungjun Kim, Jia Chen, Ying-Chen Chen, Min-Hwi Kim, Hyungjin Kim, Min-Woo Kwon, Sungmin Hwang, Muhammad Ismail, Yi Li, Xiang-Shui Miao, Yao-Feng Chang, and Byung-Gook Park. Neuronal dynamics in HfOx/AlOy-based homeothermic synaptic memristors with low-power and homogeneous resistive switching. *Nanoscale*, 11(1) :237–245, 2019. Publisher : The Royal Society of Chemistry.

[21] Ya Li, Paiwen Fang, Xihua Fan, and Yanli Pei. NiO-based memristor with three resistive switching modes. *Semiconductor Science and Technology*, 35(5) :055004, mar 2020.

[22] Abdelouadoud El Mesoudy, Gwénaëlle Lamri, Raphaël Dawant, Javier Arias-Zapata, Pierre Gliech, Yann Beilliard, Serge Ecoffey, Andreas Ruediger, Fabien Alibart, and Dominique Drouin. Fully cmos-compatible passive tio2-based memristor crossbars for in-memory computing, 2021.

[23] João H Quintino Palhares, Yann Beilliard, Fabien Alibart, Everton Bonturim, Daniel Z de Florio, Fabio C Fonseca, Dominique Drouin, and Andre S Ferlauto. Oxygen vacancy engineering of TaO x -based resistive memories by zr doping for improved variability and synaptic behavior. *Nanotechnology*, 32(40) :405202, jul 2021.

[24] 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.

[25] Abdelouadoud El Mesoudy, Gwénaëlle Lamri, Raphael Dawant, J. Arias-Zapata, Pierre Gliech, Y. Beilliard, S. Ecoffey, A. Ruediger, F. Alibart, and D. Drouin. Fully cmos-compatible passive tio2-based memristor crossbars for in-memory computing. *ArXiv*, abs/2106.11808, 2021.

[26] A Napolean, NM Sivamangai, R NaveenKumar, and N Nithya. Electroforming atmospheric temperature and annealing effects on pt/hfo2/tio2/hfo2/pt resistive random access memory cell. *Silicon*, 2021.

[27] Fabien Alibart, Elham Zamanidoost, and Dmitri B. Strukov. Pattern classification by memristive crossbar circuits using ex situ and in situ training. 4(1), June 2013.

[28] Fabien Alibart, Ligang Gao, Brian D. Hoskins, and Dmitri B. Strukov. High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. *Nanotechnology*, 23(7) :075201, January 2012. Publisher : IOP Publishing.

[29] Yang Yin Chen, Ludovic Goux, Sergiu Clima, Bogdan Govoreanu, Robin Degraeve, Gouri Sankar Kar, Andrea Fantini, Guido Groeseneken, Dirk J. Wouters, and Malgorzata Jurczak. Endurance/retention trade-off on HfO_2/Metal cap 1t1r bipolar rram. *IEEE Transactions on Electron Devices*, 60(3) :1114–1121, 2013.

[30] Jiun-Jia Huang, Yi-Ming Tseng, Chung-Wei Hsu, and Tuo-Hung Hou. Bipolar nonlinear Ni/TiO_2/Ni selector for 1s1r crossbar array applications. *IEEE Electron Device Letters*, 32(10) :1427–1429, 2011.

[31] Zheng Chai, Pedro Freitas, Weidong Zhang, Firas Hatem, Jian Fu Zhang, John Marsland, Bogdan Govoreanu, Ludovic Goux, and Gouri Sankar Kar. Impact of rtn on pattern recognition accuracy of rram-based synaptic neural network. *IEEE Electron Device Letters*, 39(11) :1652–1655, 2018.

[32] Sapan Agarwal, Richard L Schiek, and Matthew J Marinella. Compensating for parasitic voltage drops in resistive memory arrays. In *2017 IEEE International Memory Workshop (IMW)*. IEEE, May 2017.

[33] An Chen. A comprehensive crossbar array model with solutions for line resistance and nonlinear device characteristics. *IEEE Transactions on Electron Devices*, 60(4) :1318–1326, 2013.

[34] D. Joksas, P. Freitas, Z. Chai, W. H. Ng, M. Buckwell, C. Li, W. D. Zhang, Q. Xia, A. J. Kenyon, and A. Mehonic. Committee machines—a universal method to deal with non-idealities in memristor-based neural networks. *Nature Communications*, 11(1), aug 2020.

[35] Jian Kang, Zhizhen Yu, Lindong Wu, Yichen Fang, Zongwei Wang, Yimao Cai, Zhigang Ji, Jianfu Zhang, Runsheng Wang, Yuchao Yang, and Ru Huang. Time-dependent variability in RRAM-based analog neuromorphic system for pattern recognition. In *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, December 2017.

[36] Lixue Xia, Wenqin Huangfu, Tianqi Tang, Xiling Yin, Krishnendu Chakrabarty, Yuan Xie, Yu Wang, and Huazhong Yang. Stuck-at fault tolerance in rram computing systems. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(1) :102–115, 2018.

[37] Adnan Mehonic, Dovydas Joksas, Wing H Ng, Mark Buckwell, and Anthony J Kenyon. Simulation of inference accuracy using realistic RRAM devices. *Front. Neurosci.*, 13 :593, June 2019.

[38] Vinay Joshi, Manuel Le Gallo, Simon Haefeli, Irem Boybat, S. R. Nandakumar, Christophe Piveteau, Martino Dazzi, Bipin Rajendran, Abu Sebastian, and Evangelos Eleftheriou. Accurate deep neural network inference using computational phase-change memory. *Nature Communications*, 11(1), may 2020.

[39] Abinash Mohanty, Xiaocong Du, Pai-Yu Chen, Jae-sun Seo, Shimeng Yu, and Yu Cao. Random sparse adaptation for accurate inference with inaccurate multi-level rram arrays. In *2017 IEEE International Electron Devices Meeting (IEDM)*, pages 6.3.1–6.3.4, 2017.

[40] Beiye Liu, Hai Li, Yiran Chen, Xin Li, Qing Wu, and Tingwen Huang. Vortex. In *Proceedings of the 52nd Annual Design Automation Conference*. ACM, jun 2015.

[41] Lerong Chen, Jiawen Li, Yiran Chen, Qiuping Deng, Jiyuan Shen, Xiaoyao Liang, and Li Jiang. Accelerator-friendly neural-network training : Learning variations and defects in RRAM crossbar. In *Design, Automation &amp Test in Europe Conference &amp Exhibition (DATE), 2017*. IEEE, mar 2017.

[42] Michael Klachko, Mohammad Reza Mahmoodi, and Dmitri B. Strukov. Improving noise tolerance of mixed-signal neural networks. *CoRR*, abs/1904.01705, 2019.

[43] Miao Hu, John Paul Strachan, Zhiyong Li, Emmanuelle M. Grafals, Noraica Davila, Catherine Graves, Sity Lam, Ning Ge, Jianhua Joshua Yang, and R. Stanley Williams. Dot-product engine for neuromorphic computing. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, jun 2016.

[44] Chenchen Liu, Miao Hu, John Paul Strachan, and Hai (Helen) Li. Rescuing memristor-based neuromorphic design with high defects. In *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, jun 2017.

[45] Tayfun Gokmen, Malte J. Rasch, and Wilfried Haensch. The marriage of training and inference for scaled deep learning analog hardware. In *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, dec 2019.

[46] Farnood Merrikh Bayat, Mirko Prezioso, Bhaswar Chakrabarti, Irina Kataeva, and Dmitri Strukov. Memristor-based perceptron classifier : Increasing complexity and coping with imperfect hardware. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, nov 2017.

[47] Mohammad R. Mahmoodi, Adrien F. Vincent, Hussein Nili, and Dmitri B. Strukov. Intrinsic bounds for computing precision in memristor-based vector-by-matrix multipliers. *IEEE Transactions on Nanotechnology*, 19 :429–435, 2020.

[48] Z. Fahimi, M. R. Mahmoodi, M. Klachko, H. Nili, H. Kim, and D. B. Strukov. Mitigating imperfections in mixed-signal neuromorphic circuits, 2021.

[49] Yue Zhou, Xiaofang Hu, Lidan Wang, and Shukai Duan. Bayesian neural network enhancing reliability against conductance drift for memristor neural networks. *Sci. China Inf. Sci.*, 64(6) :160408, June 2021.

[50] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence : Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8) :1738–1762, 2019.

[51] H. Li, B. Gao, Z. Chen, Y. Zhao, P. Huang, H. Ye, L. Liu, X. Liu, and J. Kang. A learnable parallel processing architecture towards unity of memory and computing. *Scientific Reports*, 5(1), August 2015.

[52] Amirali Amirsoleimani, Fabien Alibart, Victor Yon, Jianxiong Xu, M. Reza Pazhouhandeh, Serge Ecoffey, Yann Beilliard, Roman Genov, and Dominique

Drouin. In-memory vector matrix multiplication in monolithic complementary metal–oxide–semiconductor-memristor integrated circuits : Design choices, challenges, and perspectives. *Advanced Intelligent Systems*, 2(11) :2000115, August 2020.

[53] Doo Seok Jeong, Kyung Min Kim, Sungho Kim, Byung Joon Choi, and Cheol Seong Hwang. Memristors for energy-efficient new computing paradigms. *Advanced Electronic Materials*, 2(9) :1600090, August 2016.

[54] Daniele Ielmini and Giacomo Pedretti. Device and circuit architectures for in-memory computing. *Advanced Intelligent Systems*, 2(7) :2000040, May 2020.

[55] Abdelouadoud El Mesoudy, Gwénaëlle Lamri, Raphaël Dawant, Javier Arias-Zapata, Pierre Gliech, Yann Beilliard, Serge Ecoffey, Andreas Ruediger, Fabien Alibart, and Dominique Drouin. Fully CMOS-compatible passive TiO2-based memristor crossbars for in-memory computing. *Microelectronic Engineering*, 255 :111706, February 2022.

[56] Thomas Dalgaty, Eduardo Esmanhotto, Niccolo Castellani, Damien Querlioz, and Elisa Vianello. Ex situ transfer of Bayesian neural networks to resistive memory-based inference hardware. *Advanced Intelligent Systems*, 3(8) :2000103, May 2021.

[57] H. Kim, M. R. Mahmoodi, H. Nili, and D. B. Strukov. 4k-memristor analog-grade passive crossbar circuit. *Nature Communications*, 12(1), August 2021.

[58] F. Merrikh Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov. Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nature Communications*, 9(1), June 2018.

[59] D. Kau, Stephen Tang, Ilya Karpov, Rick Dodge, Brett Klehn, Johannes Kalb, Jonathan Strand, Aleshandre Diaz, Nelson Leung, Jack Wu, Sean Lee, Tim Langtry, Kuo-Wei Chang, Christina Papagianni, Jinwook Lee, Jeremy Hirst, Swetha Erra, Eddie Flores, Nick Righos, and Gianpaolo Spadini. A stackable cross point phase change memory. pages 1–4, Jan. 2010.

[60] Y. Nishitani, Y. Kaneko, M. Ueda, T. Morie, and E. Fujii. Three-terminal ferroelectric synapse device with concurrent learning function for artificial neural networks. *Journal of Applied Physics*, 111(12) :124108, June 2012.

[61] Honey Nikam, Siddharth Satyam, and Shubham Sahay. Long short-term memory implementation exploiting passive RRAM crossbar array. *IEEE Transactions on Electron Devices*, 69(4) :1743–1751, 2022.

[62] Jianhui Han, He Liu, Mingyu Wang, Zhaolin Li, and Youhui Zhang. ERA-LSTM : An efficient ReRam-based architecture for long short-term memory. *IEEE Transactions on Parallel and Distributed Systems*, PP :1–1, 12 2019.

[63] Weijian Chen, Zhi Qi, Zahid Akhtar, and Kamran Siddique. Resistive-RAM-based in-memory computing for neural network : A review. *Electronics*, 11(22) :3667, November 2022.

[64] Yudeng Lin, Qingtian Zhang, Jianshi Tang, Bin Gao, Chongxuan Li, Peng Yao, Zhengwu Liu, Jun Zhu, Jiwu Lu, Xiaobo Sharon Hu, He Qian, and Huaqiang Wu. Bayesian neural network realization by exploiting inherent stochastic characteristics of analog RRAM. In *2019 IEEE International Electron Devices Meeting (IEDM)*, pages 14.6.1–14.6.4, 2019.

[65] Akul Malhotra, Sen Lu, Kezhou Yang, and Abhronil Sengupta. Exploiting oxide based resistive RAM variability for Bayesian neural network hardware design. *IEEE Transactions on Nanotechnology*, 19 :328–331, 2020.

[66] Fabien Alibart, Elham Zamanidoost, and Dmitri B. Strukov. Pattern classification by memristive crossbar circuits using ex situ and in situ training. *Nature Communications*, 4(1), June 2013.

[67] Cheng-Xin Xue, Tsung-Yuan Huang, Je-Syu Liu, Ting-Wei Chang, Hui-Yao Kao, Jing-Hong Wang, Ta-Wei Liu, Shih-Ying Wei, Sheng-Po Huang, Wei-Chen Wei, Yi-Ren Chen, Tzu-Hsiang Hsu, Yen-Kai Chen, Yun-Chen Lo, Tai-Hsing Wen, Chung-Chuan Lo, Ren-Shuo Liu, Chih-Cheng Hsieh, Kea-Tiong Tang, and Meng-Fan Chang. 15.4A 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices. *2020 IEEE International Solid- State Circuits Conference (ISSCC)*, pages 244–246, 2020.

[68] Peng Yao, Huaqiang Wu, Bin Gao, Sukru Burc Eryilmaz, Xueyao Huang, Wenqiang Zhang, Qingtian Zhang, Ning Deng, Luping Shi, H.-S. Philip Wong, and He Qian. Face classification using electronic synapses. *Nature Communications*, 8(1), May 2017.

[69] Qiangfei Xia and J. Joshua Yang. Memristive crossbar arrays for brain-inspired computing. *Nature Materials*, 18(4) :309–323, March 2019.

[70] S. Ambrogio, S. Balatti, A. Cubeta, A. Calderoni, N. Ramaswamy, and D. Ielmini. Understanding switching variability and random telegraph noise in resistive RAM. In *2013 IEEE International Electron Devices Meeting*, pages 31.5.1–31.5.4, 2013.

[71] Victor Yon, Amirali Amirsoleimani, Fabien Alibart, Roger G. Melko, Dominique Drouin, and Yann Beilliard. Exploiting non-idealities of resistive switching memories for efficient machine learning. *Frontiers in Electronics*, 3, March 2022.

[72] Z. Fahimi, M. R. Mahmoodi, M. Klachko, H. Nili, and D. B. Strukov. The impact of device uniformity on functionality of analog passively-integrated memristive circuits. *IEEE Transactions on Circuits and Systems I : Regular Papers*, 68(10) :4090–4101, 2021.

[73] Z. Fahimi, M. R. Mahmoodi, M. Klachko, H. Nili, H. Kim, and D. B. Strukov. Mitigating imperfections in mixed-signal neuromorphic circuits, 2021.

[74] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes, 2013.

[75] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4) :591–611, December 1965.

[76] Fabien Alibart, Ligang Gao, Brian D Hoskins, and Dmitri B Strukov. High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm. *Nanotechnology*, 23(7) :075201, January 2012.

[77] F. Merrikh Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov. Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits. *Nature Communications*, 9(1), June 2018.

[78] Gina C. Adam, Brian D. Hoskins, Mirko Prezioso, Farnood Merrikh-Bayat, Bhaswar Chakrabarti, and Dmitri B. Strukov. 3-D memristor crossbars for analog and

neuromorphic computing applications. *IEEE Transactions on Electron Devices*, 64(1) :312–318, 2017.

[79] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2009.

[80] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization, 2014.

[81] John Aldrich. R.A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3), September 1997.

[82] Charles Mackin, Malte J. Rasch, An Chen, Jonathan Timcheck, Robert L. Bruce, Ning Li, Pritish Narayanan, Stefano Ambrogio, Manuel Le Gallo, S. R. Nandakumar, Andrea Fasoli, Jose Luquin, Alexander Friz, Abu Sebastian, Hsinyu Tsai, and Geoffrey W. Burr. Optimised weight programming for analogue memory-based deep neural networks. *Nature Communications*, 13(1) :3765, June 2022.

[83] P. Drolet and V. Yon. Hardware-aware article library. `https://github.com/3it-nano/Hardware-Aware-Article`, 2023.

[84] P-A Mouny. B1530a control script. `https://github.com/3it-nano/B1530A_control`, 2021.

[85] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press, Cambridge, 2009.

[86] Hojin Moon, Hongshik Ahn, Ralph L. Kodell, Songjoon Baek, Chien-Ju Lin, and James J. Chen. Ensemble methods for classification of patients for personalized medicine with high-dimensional data. *Artificial Intelligence in Medicine*, 41(3) :197–207, November 2007.

[87] Scott McLachlan, Kudakwashe Dube, Graham A Hitman, Norman E Fenton, and Evangelia Kyrimi. Bayesian networks in healthcare : Distribution by medical condition. *Artificial Intelligence in Medicine*, 107 :101912, July 2020.

[88] J. M. Pena, J. Bjorkegren, and J. Tegner. Growing Bayesian network models of gene networks from seed genes. *Bioinformatics*, 21(Suppl 2) :ii224–ii229, September 2005.

[89] Rocío Díaz de León Torres, Martín Molina, and Pascual Campoy. Survey of bayesian networks applications to intelligent autonomous vehicles, 2019.

[90] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On markov chain monte carlo methods for tall data, 2015.

[91] John D. Cook. Mcmc burn-in is misunderstood, May 2020.

[92] Charles Geyer. Burn-in is unnecessary.

[93] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4) :327–335, 1995.

[94] Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler : Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *arXiv :1111.4246 [cs, stat]*, November 2011. arXiv : 1111.4246.

[95] Chuck Huber. Mcmc and the metropolis hastings algorithm.

[96] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1) :79–86, 1951.

[97] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[98] Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic variational inference, 2013.

[99] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference, 2016.

[100] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. *arXiv :1505.05424 [cs, stat]*, May 2015. arXiv : 1505.05424.

[101] Manfred Opper and Cedric Archambeau. The variational gaussian approximation revisited. *Neural computation*, 21 :786–92, 10 2008.

[102] Ioannis Messaris, Alexander Serb, Spyros Stathopoulos, Ali Khiat, Spyridon Nikolaidis, and Themistoklis Prodromakis. A data-driven verilog-a reram model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(12) :3151–3162, 2018.

[103] Sachin Maheshwari, Spyros Stathopoulos, Jiaqi Wang, Alexander Serb, Yihan Pan, Lieuwe B. Leene, Christos Papavassiliou, Timothy G. Constandinou, and Themistoklis Prodromakis. Hybrid cmos/memristor circuit design methodology, 2020.

[104] Marc Bocquet, Damien Deleruyelle, Hassen Aziza, Christophe Muller, Jean-Michel Portal, Thomas Cabout, and Eric Jalaguier. Robust Compact Model for Bipolar Oxide-Based Resistive Switching Memories. *IEEE Trans. Electron Devices*, 61(3) :674–681, March 2014.

[105] Zizhen Jiang, Shimeng Yu, Yi Wu, Jesse H. Engel, Ximeng Guan, and H.-S. Philip Wong. Verilog-a compact model for oxide-based resistive random access memory (rram). In *2014 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, pages 41–44, 2014.

[106] Dhirendra Vaidya, Shraddha Kothari, Thomas Abbey, Spyros Stathopoulos, Loukas Michalas, Alexantrou Serb, and Themis Prodromakis. Compact modeling of the switching dynamics and temperature dependencies in tio memristors—part ii : Physics-based model. *IEEE Transactions on Electron Devices*, 68(10) :4885–4890, 2021.

[107] Dhirendra Vaidya, Shraddha Kothari, Thomas Abbey, Ali Khiat, Spyros Stathopoulos, Loukas Michalas, Alexantrou Serb, and Themis Prodromakis. Compact modeling of the switching dynamics and temperature dependencies in tio-based memristors—part i : Behavioral model. *IEEE Transactions on Electron Devices*, 68(10) :4877–4884, 2021.